(4)

AAMRL-TR-87-040

INTEGRATED ANALYSIS TECHNIQUES FOR COMMAND, CONTROL, AND COMMUNICATIONS SYSTEMS (U)

Volume I: Methodology (U)

JOSEPH G. WOHL
ROBERT R. TENNEY

ALPHATECH, INC.

NOVEMBER 1987

FINAL REPORT FOR AUGUST 1982 - DECEMBER 1985

Approved for public release, distribution is unlimited.

HARRY G. ARMSTRONG AEROSPACE MEDICAL RESEARCH LABORATORY
HUMAN SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6573

NOTICES

When US Government drawings, specifications, or other data are used for any purpose other than a definitely related Government procurement operation, the Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise, as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

Please do not request copies of this report from Armstrong Aerospace Medical Research Laboratory. Additional copies may be purchased from:

National Technical Information Service
5285 Port Royal Road
Springfield, Virginia 22161

Federal Government agencies and their contractors registered with Defense Technical Information Center should direct requests for copies of this report to:

Defense Technical Information Center
Cameron Station
Alexandria, Virginia 22314

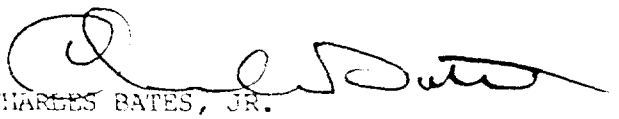
TECHNICAL REVIEW AND APPROVAL

AAMRL-TR-87-040

This report has been reviewed by the Office of Public Affairs (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER



CHARLES BATES, JR.
Director, Human Engineering Division
Armstrong Aerospace Medical Research Laboratory

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) TR-293-1 Volume I	5. MONITORING ORGANIZATION REPORT NUMBER(S) AAMRL-TR-87-040
6a. NAME OF PERFORMING ORGANIZATION *ALPHATECH, Inc. 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Harry G. Armstrong Aerospace Medical Research Laboratory - AAMRL/HED
6c. ADDRESS (City, State, and ZIP Code) 2 Burlington Executive Center 111 Middlesex Turnpike Burlington, MA 01803	7b. ADDRESS (City, State, and ZIP Code) Wright-Patterson AFB, OH 45433-6573
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-82-C-0509
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 62202F; PROJECT NO. 6893; TASK NO. 04; WORK UNIT ACCESSION NO. 63

11. TITLE (Include Security Classification)
INTEGRATED ANALYSIS TECHNIQUES FOR COMMAND, CONTROL, AND COMMUNICATIONS SYSTEMS VOLUME I: METHODOLOGY (U)

12. PERSONAL AUTHOR(S)
Wohl, Joseph G., and Tenney, Robert R.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM 8/32 TO 12/85	1987 November	191

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	system representation, system analysis, performance analysis, Petri nets, queuing networks, hierarchical decomposition
05	08		
17	02		

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

This is the first volume of a two-volume report describing research on Integrated Analysis Techniques (IAT) sponsored by AAMRL's C^3 Operator Performance Engineering (COPE) Program. At its present state of development, IAT is a comprehensive framework for the representation and analysis of C^3 systems. This framework consists of:

- A hierarchical method for describing a C^3 system along the four dimensions of process, resource, organization, and goal,
- A mathematical construct for C^3 system modeling (Stochastic, Timed, Attributed Petri Nets, or STAPNs), and
- Several C^3 system performance analysis methods (STAPNs, PERT/CPM, and queuing networks).

(Continued over)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS	UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Donald L. Monk	22b. TELEPHONE (Include Area Code) (513) 255-8814
	22c. OFFICE SYMBOL AAMRL/HED

DD Form 1473, JUN 86 *Previous editions are obsolete.*

19. Abstract (Continued)

As reported in Volume II, several trial applications of IAT were completed, which helped evolve the approach and also supported the basic validity of the framework. However, these applications did indicate that for IAT ever to become a useful analyst's tool, it must be computer-aided.

SUMMARY

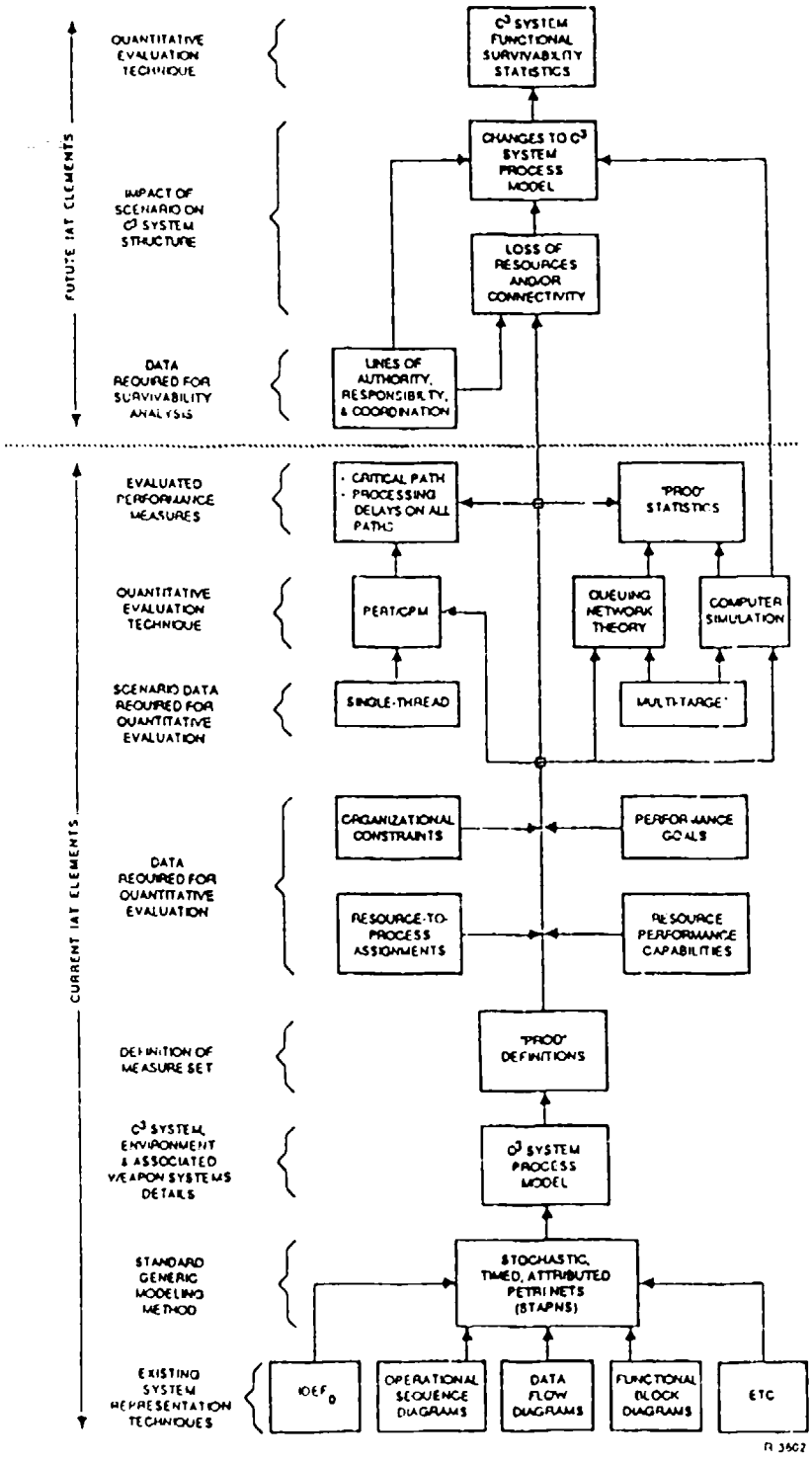
This report consists of two volumes: In Volume I, the Integrated Analysis Techniques (IAT) for Command, Control and Communications (C^3) systems are described, along with the background, concept, requisite methodologies, and recommendations for an automated analyst's aid. In Volume II, recommendations for an automated analyst's aid. In Volume 2, the evolution of IAT via successive trial applications to three (C^3) systems or subsystems is described and the lessons learned are summarized.

This first volume summarizes the results achieved to date. These results clearly indicate the feasibility of IAT, as well as the relationships with existing techniques (e.g., DeMarco Data Flow Diagrams, $IDEF_0$, Operational Sequence Diagrams, simulation languages, etc.). In particular, the following results are described:

- A four-dimensional analytic framework for IAT, along with a definitive set of requirements to be met;
- A symbolic language involving a major extension of Petri net theory, for modeling and evaluating the performance of manned C^3 systems at any level of description or decomposition;
- A convenient means for aggregating and modularizing system details without masking their impact on system performance;
- A set of nested, self-consistent and upward-aggregatable system performance and effectiveness measures derived directly from the symbolic language;
- A set of rules for applying the overall methodology;
- A flexible database management approach to building and storing the requisite model structure and data; and
- Recommend features for an automated analyst's aid to applying IAT to manned C^3 systems,

Details of the methodology and guidelines for their application are described in a series of Appendices.

The accompanying figure summarizes the relationship among IAT elements and also indicates both progress and areas of future work.



Relationship Among IAT Elements

PREFACE

This work was conducted by personnel of ALPHATECH, Inc. under contract F33615-82-C-0509 with the Harry G. Armstrong Aerospace Medical Research Laboratory, Human Systems Division, Air Force Systems Command, Wright Patterson Air Force Base, Ohio. The methods summarized in this report were developed under program 62202F, Aerospace Biotechnology, Project 7184, Man-Machine Integration Technology.

The authors wish to acknowledge the important contributions to this report made by several people. Dr. David L. Kleinman, of the University of Connecticut, acted as consultant in developing many of the ideas incorporated in Section 2. Dr. John J. Shaw of ALPHATECH contributed to Section 3. Dr. Richard A. Miller of Ohio State University acted as consultant in analytic methods and prepared parts of Section 4. Dr. Judith R. Kornfeld contributed to Appendices C and D.

In addition to those individuals mentioned above, the authors would like to thank Mr. James C. Deckert and Dr. Nils R. Sandell, Jr. of ALPHATECH, Mr. Maris Vikmanis and Mr. Donald Monk of AAMRL, and Major Richard Poturalski of Headquarters, Air Force Space Command for their continuing encouragement and support in the face of sometimes seemingly insurmountable obstacles.

CONTENTS

CONTENTS (Continued)

CONTENTS (Continued)

CONTENTS (Continued)

FIGURES

FIGURES (Continued)

TABLES

SECTION 1

INTRODUCTION

1.1 PURPOSE OF THIS STUDY

This three-year study was undertaken to begin the development of a set of Integrated Analysis Techniques (IAT) for deriving quantitative measures of the performance and military effectiveness of Command, Control, and Communications (C^3) systems. The approach taken was to study several C^3 systems (or portions thereof) in depth; to develop a method for representing these systems (i.e., accurately describing their subsystems, performance parameters, interrelationships, human activities, and military effectiveness); to apply the method to the selected systems; and to codify the method so that other analysts could apply it as needed to other systems. This report summarizes the study results.

The developmental results to date clearly indicate the feasibility of IAT as well as the relationships with existing techniques (i.e., DeMarco data flow diagrams, $IDEF_0$, operational sequence diagrams, simulation languages, etc.). Trial applications have resulted in critical "lessons learned" that are also presented here.

In a word, the main obstacle to integration of the many representational techniques has been the lack of a single underlying analytical framework (i.e., a Theory of C^3) which at once could (1) support quantitative performance evaluation, (2) be used at any level of system description, (3) utilize inputs obtained from any other representational method (e.g., one most familiar to the user), and (4) represent C^3-specific system characteristics such as hierarchical organization structure and the means for system adaptability and survivability in the face of enemy attack.

1.2 GENERIC C^3 ANALYSIS ISSUES

The following subsections highlight the current issues in analyzing C^3 systems: (1) System representation and modeling issues; (2) How C^3 systems differ from other complex large-scale system; (3) Human-related issues in C^3 systems; and (4) Assisting the decisionmakers within C^3 systems, and assisting C^3 analysts who are analyzing, designing or redesigning C^3 systems.

1.2.1 System Representation and Modeling

In almost all quantitative systems engineering analysis, the usual starting point is the development of a mathematical model to represent the system under investigation or development. Such a model is essential to obtaining both a precise understanding of system function and structure (i.e., architecture) and a quantitative evaluation of system performance. However, for large-scale, complex systems, a single "super-model" or set of equations is generally impossible to develop without first decomposing the system into a number of submodels. The performance characteristics of these lower-level models can then be derived quantitatively, and the results aggregated bottom-up into overall performance measures for the system. Computer simulation is often employed both to embody the lower-level models and to compute the desired measures.

For extremely complex systems, however, the modeling process usually begins with a more limited objective, namely, finding a graphic way to represent system structure and function (i.e., system architecture). This is a first step prior to any attempt at quantitative analysis. It is not uncommon for the analyst to go through the following stages in evolving a graphic representational scheme:

1. First, he develops an understanding of what the system is and how it works. Critical to such an understanding is a means of "visualizing" the system, its parts, its boundaries, and its functions. To help in the process of visualization, he may draw diagrams to represent the subsystems and their functional interrelationships. He may also develop several decomposition levels of such diagrams, in order to indicate successively more detailed understanding and to provide a basis for later detailed mathematical modeling.

2. Second, he tries to communicate this understanding to others, usually via his diagrams. He immediately finds that the same diagram can mean different things to different people, reflecting differences in their background and experience.

3. He then searches for a more or less standard (or at least well-accepted) visualization method (e.g., $IDEF_0$ functional block diagrams, DeMarco data flow diagrams, etc.) and attempts to translate his original diagrams into the new form.

4. He may find things in his original representation that are difficult to translate into the new form, and may need to invent modifications to the standard method to represent these exceptions.

5. He may try to gain peer acceptance for the "new" or "modified standard" method.

1.2.2 Differences Between C^3 and Other Large-Scale Systems

The foregoing approach generally works until a new class of system is encountered for which the new method is inadequate. While it has proven quite effective for selected large-scale systems such as power distribution systems (Shaw and Bertsekas, 1985) and large electronics maintenance facilities (Pattipati et al., 1984), it has not worked well for complex military C^3 systems. Indeed, for the past six years, the problem of system modeling and representation has been among the most important focal points for the Annual Conferences on Command, Control and Communications sponsored jointly by the Massachusetts Institute of Technology and the U.S. Office of Naval Research.

One might well ask why this is so. The main reason is that there seem to be major differences between military C^3 systems and other large-scale systems. These differences appear to be of both degree and kind. First, C^3 systems differ in degree because they are generally more geographically extensive, more complex, and involve interactions among more different types of subsystems as well as humans. Examples include the North American Aerospace Defense System, the Tactical Air Control System, and the current conceptual development of a Battle Management/C^3 System for the new U.S. Strategic Defense Initiative.

More importantly, however, C^3 systems differ in kind from other large-scale systems. Specifically:

1. Their performance is measured in terms of their contributions to an offensive or defensive military mission rather than as an end in itself;

2. Rather than having to meet a single performance goal (e.g., end-to-end message delay, units produced per unit time), they must be capable of meeting multiple and even conflicting goals (e.g., maximize enemy aircraft engaged per unit time while minimizing fratricide). They must also be able to adapt to changes in the military situation as required.

3. They must be able to survive deliberate enemy attacks against them in addition to responding to normal internal system degradation and failures;

4. They must exist and function within a rigid, hierarchically-structured military organization.

Finally, whereas the analysis of systems such as manufacturing and inventory control systems, pure communications systems, and management information systems involves consideration of either information quantities such as messages or physical quantitities such as manufactured items, analysis of C^3 systems involves both, and in a very special way. In effect, a race occurs between the information quantities and the physical quantities in the system. For example, as shown in Fig. 1-1, in a Strategic Defense System target detection, identification and weapon allocation messages must all be generated and

reach their appropriate destinations before the attacking missiles can carry out their missions of destruction. In addition, while the time available for data flow in the C^3 system is determined by enemy action (i.e., it is scenario-driven), the time required for defense system response is determined by a combination of C^3 and weapon system capabilities. These include (1) data flow rates and decision delays in the C^3 structure and (2) weapon activation, response and flight times for the weapon system.

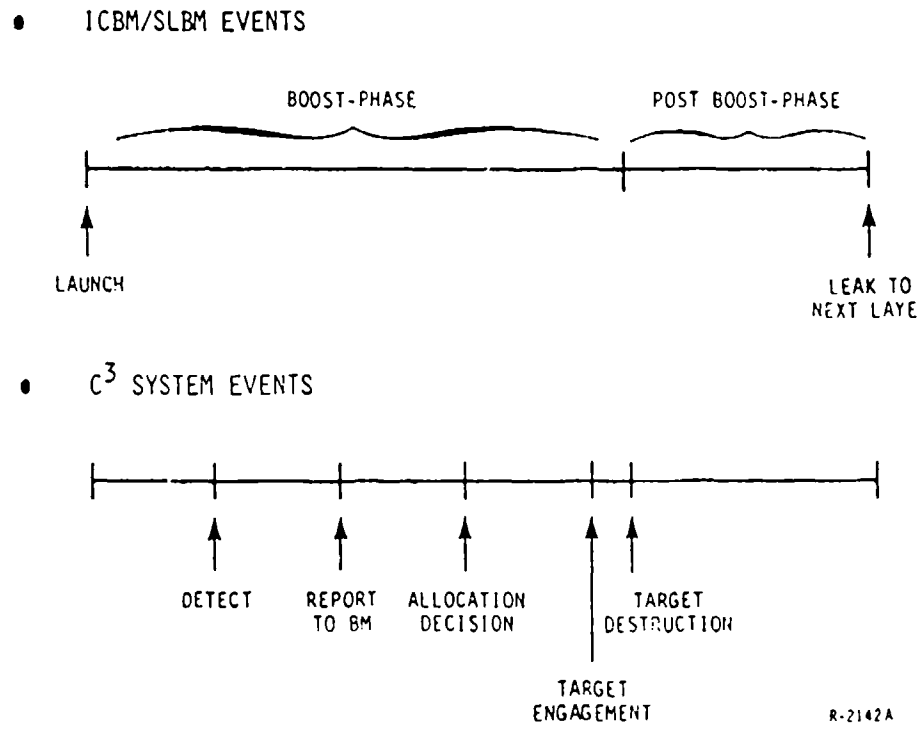


Figure 1-1. ICBM/SLBM Versus C^3 System Race

1.2.3 Human-Related Issues in C^3 Systems

It is important to note the multi-faceted nature of the roles played by humans in C^3 systems. They may function as communicators, equipment operators, or decisionmakers (and sometimes as all three simultaneously). More important, however, is the fact that wherever a human exists in a system, he/she not only represents a physical resource but also carries out a function or process while meeting the authority/responsibilty requirements and also the assigned goal of an organizational element.

It is this very fact which provides the flexibility and adaptability, and also contributes significantly to the functional survivability of C^3 systems. As organizational elements, humans can reassign goals, processes, resources and organizational responsibilities to other humans or to other mechanisms to improve overall system performance or to help reconstitute a partially destroyed system.

Note also that human performance itself is a dependent variable, affected by many system design parameters as well as by other people in the system and by specific threat and environment characteristics.

Regardless of role, human activities must be represented or modeled, as well as measured, in ways which are compatible with the models and measures of other system components and functions, in order that self-consistent aggregate system performance measures may be obtained; this has been the source of major difficulties for systems analysts and engineers in the past.

> A current example of this need is taken from the SDI program. A critical factor in the ability of the proposed Space Defense System to meet its ballistic missile "shield" objective is its ability to detect and kill attacking missiles while they are still in their boost phase. While the time required between detection and weapon assignment can be minimal, weapon release will depend on the intervention of human decisionmakers. The time available for decision will be completely circumscribed by the time between booster launch detection and re-entry vehicle deployment, and the decision itself will be further complicated by such factors as raid size, probable targets, and intelligence information. For this reason, alternate "rules of engagement" or defensive modes for the system must be developed long before it is actually employed, with defense selection being done in near-real-time based on the kinds of factors mentioned above.

The implications of the foregoing facts for C^3 system design represent an additional set of human-related issues: How should certain components of such systems be designed so as to assist individual humans as well as teams of humans in their various roles and tasks in order to improve their performance as system components? To make the best use of their special capabilities? To counterbalance the effects of human limitations?

1.2.4 The Requirement to Help Decisionmakers

We must now distinguish between two fundamentally different decisionmakers. There are those who are imbedded within a C^3 system, such as the weapon release decisionmaker in the SDI example given above, or the identification officer in an air defense system. Clearly, these individuals perform critical tasks involving situation assessment and target discrimination. These tasks may require varying degrees of assistance, depending upon such factors as time available, degree of expertise, task complexity, and so forth. The kinds of issues noted in the preceding subsection are directly relevant to imbedded decisionmakers.

However, we must also understand the needs of another class of decisionmakers, namely those who analyze and design C^3 systems. As described earlier in this section, the problems of system representation and modeling, of measurement, and especially of tracing human contributions to system performance and effectiveness have become sufficiently complex and critical that new tools are needed to help C^3 systems analysts and designers in doing their jobs.

1.3 THE IAT QUESTIONS

In recognition of (1) the needs of systems analysts for new analysis tools, (2) the differences between C^3 and other types of systems, and (3) the impact of these differences on the requirements for C^3 system representation and modeling, a set of critical questions was posed in early 1982 by Mr. M. Vikmanis and Capt. R. Poturalski of the Harry G. Armstrong Aerospace Medical Research Laboratory. Slightly paraphrased and reordered to improve clarity, the questions are as follows:

1. Given a static structural description of a C^3 system, how can one determine the system's performance?

2. What can the static structural description tell one about:

   - the strengths and weaknesses of the way functions are performed (i.e., by the mechanisms or resources which carry out the functions)?

   - the strengths and weaknesses of the way functions are combined (i.e., carried out by the same resource)?

   - the dependency of functions (i.e., upon other functions, resources, etc.)?

   - the strengths and weaknesses of data flows and controls (i.e., functional connectivity)?

   - the criticality of functions, data flows, mechanisms, and controls?

3. How can one use a static structural description, along with any other transformations, augmentations, or other data, to answer the questions in 2 above? What measures can be used?

4. What can the static structural description tell us about the dynamic performance of the system? How does it address or support issues of:

   - timeliness

   - probability of error

   - survivability?

5. What do classical systems engineering theory, organization theory, or network theory offer in the way of properties or measures to address the foregoing issues?

6. How can the answers to the above questions be used to improve system performance?

These original IAT questions provided the impetus for a 1982 study entitled "Integrated Analysis Techniques (IAT) for Application to Command, Control and Communications Systems" (Colter et al., 1982), whose results are summarized below. (The degree to which these questions can now be answered will be discussed in subsection 3.9.)

1.4 SUMMARY OF PREVIOUS STUDY RESULTS

While the direction for the 1982 study was based on the representational capabilities of the $IDEF_0$ methodology (see Section 2 for a summary description), its stated objective was "...to examine additional analysis and evaluation procedures in order to address the above issues." (Colter, et al., 1982, p. 1).

The study clearly indicated the inadequacy of $IDEF_0$ by itself to provide anything other than a skeletal structure on which to build improved analysis tools. Only limited quantitative analyses or measures are possible from an $IDEF_0$ representation of a C^3 system, regardless of the level of decomposition to which it is carried. In addition, the $IDEF_0$ symbology would have to be expanded to include such "new" standard functions as data stores.

Most important, however, was the conclusion that "...the choice of the technique(s) to be used therefore depends on the desired measurements/ characteristics, the information needs of the techniques, and the existing knowledge base," (Colter et al., 1982, p. 170).

The most significant results of the 1982 study are summarized in Table 1-1. This table identifies the specific C^3 system measures considered, the various analysis techniques applicable to them, the ability of each technique to provide quantitative versus qualitative analysis, and whether or not such analysis requires additional information beyond that embodied in the analysis technique itself.

The study concluded that the tools needed for C^3 system analysis will depend on the desired measures to be taken on the system; that no single existing technique can provide for more than a few such measures; and finally, that while $IDEF_0$ with suitable modifications can usefully support other types of analyses as well as certain direct qualitative analyses (e.g., tracing functional connectivity), it cannot provide any quantitative measures by itself.

In another study (Bachert et al., 1981) an attempt was made to use $IDEF_0$ as a "front-end" to SAINT, a simulation language developed specifically for simulating manned systems (Chubb, 1981). However, while $IDEF_0$ could provide much of the connectivity and precedence information needed to structure the simulation, the quantitative data about processes and assigned resources, necessary to complete the simulation, could not be extracted and had to be separately developed.

TABLE 1-1. RELATIONSHIP OF ANALYSIS PROCESSES TO MEASURES/CHARACTERISTICS
(Colter et al., 1982, p. 169)

PROCEDURE \ MEASURE	EFFECTIVENESS	SURVIVABILITY	EFFICIENCY	TESTABILITY	CRITICALITY	MODIFIABILITY	COMPLEXITY	CONNECTIVITY	SPAN OF CONTROL	COUPLING	COHESION	WORKLOAD	REDUNDANCY	RESOURCE AVAILABILITY	INFORMATION QUALITY	TIMELINESS	ACCURACY	CONFIDENCE (BELIEVABILITY)	COMPLETENESS	COMMUNICATIONS LOAD	ATTENTION ALLOCATION	SKILL DISTRIBUTION	BACKGROUND KNOWLEDGE	CONGESTION	OVERLOADING	RESOURCE HOLDING TIME	PARALLEL PROCESSING	PARALLEL TASKING	INFORMAL CONNECTION
IDEF$_0$ EVALUATION					•					X	X		X						X				X						
DESCRIPTIVE MATRICES					X			X		X	X	X								X				X	•				
FAMILY TREES					X			X	X	X																			
GRAPHS		X		X	X								X																
ISM PROCEDURES					X																		X						
IDEF$_0$ PLUS DATA STORES		•	•		•	•	•	•		•	•		•	•		•				•									•
FAMILY TREES PLUS LOGIC		•	•	•	•	•	o	•	•	•	•																•	•	•
ANALYTICAL STRUCTURES	•	o	X		X	•	o	o	o	X		o	X	o	•	o	o	•	•	o	X	X	X	o	X	o	X	X	•
DATA STRUCTURES			•		•			•		•			•	•					•										
SHOR																			•					•					
EVENT ANALYSIS	X	X			X									•		•			X	X	X			X	X	X	X		
TEST MATRIX ANALYSIS				o														o	o										
SURVIVABILITY ANALYSIS		o				o	o	o					o																
ENTROPY/INFO ANALYSIS			•				•	•	o	X			o						X					•					

LEGEND: X = Process supports non-quantitative statements with existing information.
• = Process supports non-quantitative statements with additional information.
o = Process supports quantitative statements with existing information.
Δ = Process supports quantitative statements with additional information.

In effect, then, we may conclude that at the outset of the present study, there was no single, existing, available Integrated Analysis Technique that could provide answers to all of the IAT questions listed in the preceding subsection. Furthermore, there were no techniques available to meet these needs.

1.5 CURRENT STATUS

The work reported on in this report completes two of the four stages in IAT development. The first stage was the formulation of a static description methodology applicable to any system to be analyzed. The second stage was the development of quantitative techniques for estimating critical system performance parameters. The third stage, being initiated under separate subcontract, is to imbed the descriptive methodology developed to date in an easy-to-use job aid which will insure, to the extent possible, both completeness and consistency of system description and analysis. This will also provide a "quick-look" capability for quantitative performance prediction at any given level of system description or decomposition. The final stage will involve the capability for analyzing the dynamic reconfigurability of a C^3 system and the allocation of resources to functions.

1.6 CONTENTS OF THIS REPORT

The objective of this report, then, is to show how to describe a complex C^3 system in such a manner that subsequent system analysis and performance questions can be answered quantitatively. Our focus has been on how one should describe an existing system (as opposed to designing a new system), so as to capture the critical attributes of the system in ever-increasing (hierarchical) detail. However, we feel that the methodology will be equally useful in evaluating the design of new systems.

Section 2 of this volume summarizes the requirements for a static representation technique, while Section 3 describes the new hierarchical method for C^3 system description and decomposition. Section 4 describes quantitative analytic tools for use with the static description. Finaliy, Section 5 provides a summary of the results to date and a set of recommendations for completing the development of Integrated Analysis Techniques for C^3 systems, with special reference to the role of humans in these systems. Volume II of this report presents the results of separate applications of portions of the method to a simulated system and two actual systems, and the lessons learned from these applications.

SECTION 2

REQUIREMENTS FOR A HIERARCHICAL METHOD FOR STATIC SYSTEM DESCRIPTION

2.1 INTRODUCTION

The primary objective (and successful result) of IAT development since 1982 was to find a single, self-consistent way to describe and model a complex C^3 system in such a manner that manned system analysis and performance questions could be answered quantitatively. The initial focus was on how one should describe an existing system (as opposed to designing a new system). The approach taken was to (1) capture the critical attributes of the system in ever-increasing (hierarchical) detail and (2) overcome some of the major problems summarized in the 1982 study noted in the introduction.

In this section we summarize the requirements which must be met by a static representation and modeling technique in order to be able to answer the IAT questions in Section 1. These include the descriptive dimensions for C^3 systems; their decomposition requirements; the relationships among the dimensions; the types of system measures to be evaluated for a system; and the data management requirements for IAT.

2.2 FOUR DIMENSIONS FOR DESCRIBING C^3 SYSTEMS

For the simplest of systems, straightforward representations of physical composition and connectivity such as system block diagrams, engineering drawings, "exploded" views, and "family trees" have long been generally accepted techniques. For more complex systems, so-called functional description techniques such as functional block diagrams (Goode and Machol, 1957) and Data Flow Diagrams (DFDs) (DeMarco, 1979) were developed as aids to diagnosing system failures and to designing new systems. Systems involving human operators and decisionmakers exhibited special requirements for representing information flow, display, control and workplace design implications; and techniques such as the Operational Sequence Diagram (OSD) were developed to meet these needs (Brooks, 1960).

With the advent of computer software, such process charts as data flow and operational sequence diagrams evolved into the more or less standard methodology of the programming flow chart. However, as programs themselves became increasingly complex, new graphical representation techniques were developed for the analysis and design of complex software systems. The Structured Analysis and Design Technique (SADT) is an example of such attempts at managing software complexity through "software engineering" (SOFTECH, 1978).

More recently, extension of these techniques to manufacturing systems and to information systems description has required that the physical resources needed to support the various functions and processes be incorporated directly into the description (e.g., as in the Integrated Computer-Aided Manufacturing Definition Language ($IDEF_0$) as developed for the U.S. Air Force (SOFTECH, 1981).

However, when dealing with a complex, large-scale system such as a C^3 system consisting of a large collection of hardware and people performing highly interrelated and interdependent functions, we find that organizational issues and constraints transcend both the physical (resource) and functional (process) characteristics of such systems and must enter prominently into the methods for description and analysis. As an example, an Air Defense C^3 system will exhibit longer response times to attacking enemy aircraft if information flow must follow strict hierarchical reporting paths than if the organization permits cross-telling of tracks. Classically, methods of organizational description and analysis have evolved separately from (although they are often confused with) those of process description and analysis. They include organization charts which display lines of authority, responsibility, and coordination as well as methods of representing organizational dynamics (Beer, 1959; Berne, 1963).

Finally, while all systems are in some sense goal-driven, the most complex of these (including C^3 systems) involve "organizations of organizations" of subsystems and people and are characterized by a complex, interrelated and dynamic hierarchy of goals which must be explicitly accounted for in system description and analysis. Methods of goal decomposition and diagrammatic representation have evolved for such complex, large-scale systems (Warfield, 1973).

On the basis of problems encountered in applying earlier methods for system static description (e.g., $IDEF_0$, OSDs, DFDs, etc.) and considering the unique requirements imposed by military C^3 systems noted in the preceding subsection, we conclude then that the following four distinct dimensions are required to describe such systems adequately at varying levels of detail:

- Resource (physical mechanism, human, geographic location, node)
- Process (function, procedure, algorithm)
- Organizational element (subdivision, unit, individual)
- Goal (intent, performance objective)

Finally (and of critical importance), the descriptive methods ultimately developed must be cable of generating important measures of system capability.

2.3 C^3 SYSTEM DECOMPOSITION REQUIREMENTS

In the preceding subsection we defined the four dimensions along which C^3 systems must be described in order to capture their complexity as well as

their critical attributes. In this subsection we summarize the requirements for system decomposition.

It is important to note that the very concept of decomposition implies a hierarchical set of relationships within each of the four dimensions identified above. One of the major requirements of IAT is that it contains a descriptive methodology capable of representing not only the decomposition within but also the interrelationships among the four dimensions while maintaining concordance among the hierarchical levels of decomposition/detail.

Another critical requirement is that each dimension must undergo recursive decomposition, starting at some initial or highest level of minimum detail. (The notion of a system boundary is implied here, at least for purposes of analysis and/or design.) The decomposition hierarchy can then be viewed as a "tree," with the "trunk" representing the highest level and the "branches" constituting each succeeding level of greater detail. For analysis purposes, we define the "leaf" level as the point at which the decomposition is terminated and a model is used for the process representation. The model parameters and characteristics are then determined by the requirements and constraints set by the physical resource and organization entities and by the goal hierarchy to which the system is responding.

The IAT methodology requires that at any decomposition level, models must be capable of being defined and exercised in order to be able to estimate system performance and effectiveness. At the higher levels (less detail) these models must, of necessity, be extremely aggregated. This is well reflected in past attempts to represent entire C^3 systems by simple time delays. However, the requirement for recursive decomposition means that the modeling method must be "generic," that is, identically applicable at each level of decomposition, with only the amount of detail and the parameter values changing between levels. Thus, selection of the most appropriate models and definition of their interdimensional relationships (as, for example the effect of a given resource on the process that it supports) has been a major requirement for successful IAT development.

In the following subsections we define in further detail the four decomposition dimensions and the manner by which successive levels of detail must evolve. Note that the ability to decompose a system along any of its dimensions separately will be essential for C^3 systems analysis and synthesis.

2.3.1 Process Decomposition

Included in the process dimension are such things as functions, processes, procedures, protocols, and scripts. This dimension has long been recognized and used as the most salient dimension for decomposition, and has received the greatest attention in earlier efforts (SOFTECH, 1981). The hierarchical organization of a C^3 system, that is, its subdivision into subsystems (e.g., surveillance, weapon control, etc.) is directly reflected in process decomposition. While sometimes referred to as system organization or even system

structure, in this report we will use the term "process decomposition" throughout, since "organization" usually refers to the representation of authority, responsibility and coordination; while "structure" refers to the function connectivity among system resources. (See further discussion of these terms in subsection 2.3.3 below.)

Thus, the primary decomposition of a process P^L at a level L>0 involves specifying the following:

1. What higher-level process P^{L-1} it is a part of; and

2. What lower-level processes $\{P^{L+1}\}$ it consists of.

It is also necessary to specify its functional connectivities, that is, to specify from which processes at the same level a given process receives inputs, as well as to which other processes it provides outputs.

2.3.2 Resource Decomposition

Included in the resource dimension are the physical resources, equipments, nodes, locations, and physical connectivities which support or carry out the processes. The resources of the system include both its physical facilities and hardware (including imbedded computers and associated software) as well as its human operators and decisionmakers.

The decomposition of a resource into its component parts is usually well-defined from the standpoint of its physical composition. Thus a resource R^L at level L>0 (e.g., an aircraft) is part of a larger resource R^{L-1} (e.g., a squadron) and itself consists of a set of subresources or components $\{R^{L+1}\}$ (e.g., its engine, avionics, etc.) at level L+1. We make the following assumptions:

A1. Resource decomposition is nonoverlapping, i.e.,

$$R^L_i \cap R^L_j = 0 \quad , \quad i \neq j$$

Physically, this means that if R^L_i is removed (from R^{L-1}), the R^L_i $i \neq j$ remain intact (even though they may not function). And, of course, since the sum of the parts equals the whole,

A2. A human resource is nondecomposable*. Hence, if

*In some previous work, various human attributes have been defined as separately available (sub)resources. However, we believe this to be erroneous. While an individual can perform several tasks in an apparently simultaneous manner, it is clear that the apparent simultaneity is really the result of "chunking" of data, efficient time-sharing among the several tasks, and well-trained response organizations. We will assume that the human, as an operator or decisionmaker, is only capable of acting as a serial processor.

$\cup_L R^L$ is a person, then $R^L{}_{i+1} = R^L{}_i$

A3. In the case of computers, we will assume that the hardware memory elements, disks, etc. are resources. However, the programs can be either processes or resources, depending on the application (e.g., imbedded computers, which are part of a fire control system, use programs that are part of the imbedded computer resources).

In a few instances, the decomposition of a resource can be non-unique, as for example when there is no a priori logical way to group the parts that comprise the whole. To minimize non-uniqueness, we establish a linkage between resource and process decomposition by assuming that:

A4. The subresources $\{R^{L+1}{}_i\}$ at level L+1 must be those that are assigned to support or carry out the subprocesses $\{P^{L+1}{}_j\}$ at the same level of decomposition.

In existing systems, it is usually the case that resources and processes are more or less directly related (in the sense that specific resources were selected to support specific processes); thus assumption A4 will generally be satisfied. However, it is important to recognize the fact that while a given resource is assigned to support only one process, it may be capable of supporting several. The notions of flexibility and adaptability derive in large part from the ability of organizational elements in a C^3 system to reassign responsibilities among lower-level organizational elements and to reassign resources to support other processes, as will be discussed below.

2.3.3 Organizational Decomposition

The military organization provides the fundamental control mechanisms whereby humans and machines work to attain objectives. Decision authority, responsibility, coordination and goal-setting are the primary attributes associated with organizational elements; they allow decisionmakers to reassign resources, processes, and organizational elements and to modify objectives if necessary, in order to adapt to a variety of changing circumstances in the military environment.

Of course, all C^3 systems will evolve and change as they take advantage of new technology and as they are called upon to support new or changing missions (e.g., search and rescue). Also, one may be interested in questions relating to off-nominal performance such as system survivability, adaptability, and flexibility (e.g., due to loss of a resource, failure to meet an objective, etc.). In either case, one must include organizational representation in system description.

The primary decomposition of an organizational element O^L defines the lines of decision authority (i.e., command structure and accountability, or reporting-to), responsibility (i.e., control), and coordination. With respect to authority, element O^L at level L is only accountable to a single element O^{L-1} at a higher level and has authority over the set of elements $\{O^{L+1}\}$ at the next lower level L+1, which in turn are accountable to O^L. This authority decomposition defines an organizational hierarchy.

With respect to responsibility, the relationships are more complex. Organizational elements have responsibility for the processes and resources assigned to them, and authority over lower-level elements; and they are accountable to higher-level elements, in strict accordance with the lines of authority described above. Note that a human being is a resource which can fulfill various organizational requirements at different times and to different purposes or goals. Thus, there may be instances in which a lower-level element is accountable to one higher-level element for one set of processes and/or resources and to a different element for another set. Such accountability to different "bosses" for different activities characterizes the so-called matrix organization, and is exemplified by the "multi-hattedness" of many U.S. military commands. This can be quite different for non-U.S. commands.

Note also that an organizational element has responsibility for controlling processes, and that these processes can include reassignments of lower-level decision authority, i.e., of accountability and control.

Finally, from a decomposition standpoint it is important to recognize that an organizational element at level L may be responsible for two fundamentally different classes of processes:

1. those at the same level L that directly support the functions of the organizational element in question; and

2. those at the next lower organizational level L+1 into which the function at level L decomposes.

Classically, these are known as "staff" and "line functions," respectively.

As noted earlier, it is extremely important to distinguish between organizational and process decomposition. Among systems engineers the term "organization" is usually taken to refer to the way in which the system itself is hierarchically and/or functionally organized (i.e., structured or decomposed), whereas among human factors specialists, the very same term is used to represent the lines of authority, responsibility and coordination among the personnel in the system. For example, from an engineering standpoint, a C^3 system is generally "organized" into a surveillance subsystem, a planning subsystem, a controlling subsystem, an order dissemination subsystem, a communications subsystem, etc. On the other hand, the structure of the system is usually taken to refer to the functional connectivity among the resources of the system, that is, which processes must "talk to" or coordinate with which other processes, and which resources must be connected to each other in order to provide for the required data flow among the processes.

In this report we shall use the following definitions of these terms:

- Process decomposition: hierarchical subdivision of processes into their component subprocesses. Results in multiple levels of description at successively finer detail.

- Resource decomposition: hierarchical subdivision of resources into their component parts. Also results in multiple levels of description at successively finer detail.

- Organization, or organizational decomposition: hierarchical subdivision of decision authority, responsibility, and coordination among all system processes and/or resources (including humans as resources).

- Structure: connectivity among all system processes and/or resources (including humans as resources), at a given level of decomposition or description.

In some cases, as we shall see, the very existence of connectivity between two resources implies either authority (as for example in a precedence relationship such as "A before B") or coordination (as for example in a joint presence relationship such as "A and B before C"). However, the concept of responsibility is peculiarly human in that it can only be offered (to be either accepted or rejected by an individual) but never delegated: whereas authority can indeed be delegated. On the other hand, authority and coordination can always be "wired in" to a system by design, but hardware or software parts of the system cannot take responsibility for their performance. This is especially important from a human factors standpoint, since the system must be carefully designed to support the human roles (i.e., their authority, responsibility and coordination needs) in the system.

2.3.4 Goal Decomposition

Goals are established by organizational elements and must therefore be included in parallel with the organizational dimension. In fact, the establishment of goals, resolution of conflicting goals, partitioning of goals into subgoals, and assignment of responsible organizational elements to achieve these subgoals are perhaps the most important of all organizational activities (i.e., management).

The fact that an individual can and does act both as an organizational element and as a resource can be a major source of confusion unless it is recognized that:

- an organizational element sets goals for lower-level elements. This is based on the effectiveness requirements placed on level O^L by the next higher level O^{L-1};

- these goals are equivalent to performance requirements for the processes P^{L+1} at level L+1 for which O^{L+1} has responsibility (i.e., is assigned);

- the processes $P^{L+1}{}_i$ are supported by resources R^{L+1} at level L+1. The degree to which a goal G^{L+1} is met is determined by the degree to which the resources R^{L+1} assigned to that process can meet the process performance requirements implied by the goal;

- the goal becomes the means by which the entire C^3 entity (process, supporting resources, and responsible organizational element) is controlled.

Figure 2-1 summarizes the interactions among the C^3 dimensions and demonstrates how goals are used in controlling both processes, resources, and organization . Only line relationships are shown in order to simplify the diagram; staff elements would be shown as L-level processes and resources that directly support the line functions at that level.

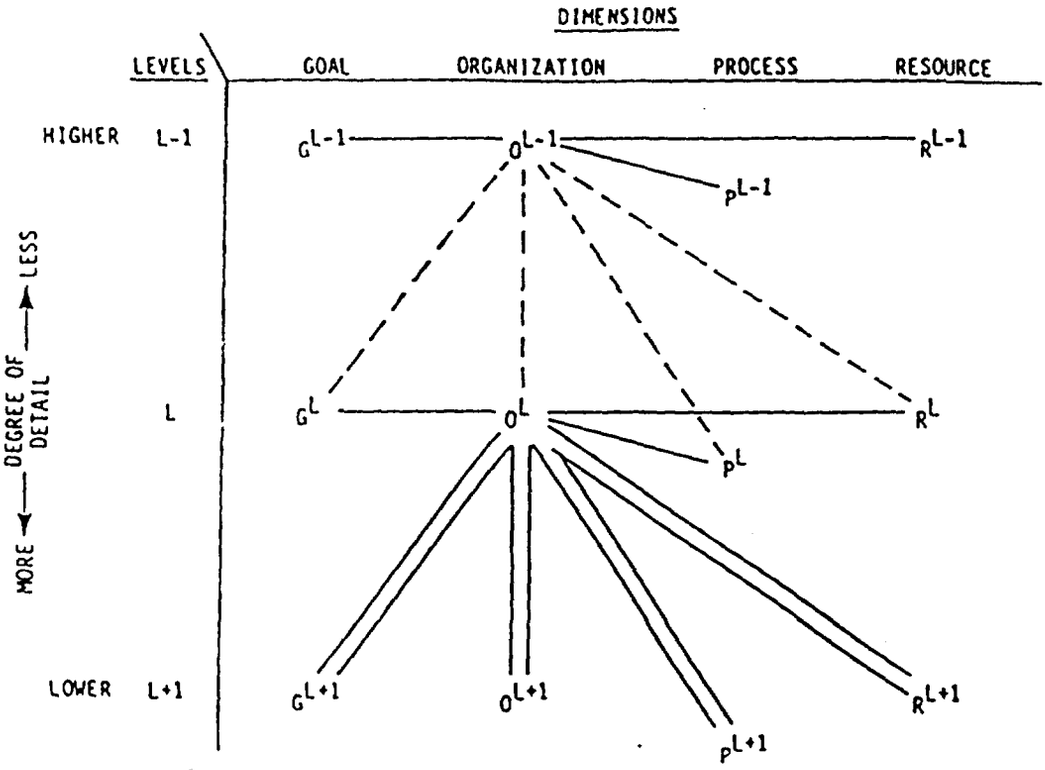
2.3.5 Relationships Among Processes, Resources, Organizational Elements, and Goals

From Fig. 2-1 it is clear that a C^3 system exhibits several classes of interactions. In this section we examine the most important of these interactions.

In general, a process requires inputs and provides outputs. The resulting information flow among processes is only an indirect determinant of system topology (i.e., it only implies connectivity); however, in reality, information exchange takes place not among processes but among the physical resources that support and are assigned to the processes. If connectivity between two C^3 resources is broken (e.g., a radio link is jammed), then the information flow between the processes supported by these resources (e.g., intelligence information) is halted. We therefore incorporate system topology in resource decomposition by defining at level L the resource connectivity matrix, $[R^L{}_i \times R^L{}_j]$;

$$[R^L{}_i \times R^L{}_j] = \begin{cases} 1 & \text{if resource } R^L{}_i \text{ sends information to resource } R^L{}_j \\ 0 & \text{otherwise} \end{cases}$$

Thus, the nonzero elements of the i-th row of $[R^L{}_i \times R^L{}_j]$ indicate those resources to which $R^L{}_i$ sends information, and the nonzero elements of the i-th column of $[R^L{}_i \times R^L{}_j]$ indicate those resources from which $R^L{}_i$ receives information. Note that the nature of connectivity implies that for every nonzero $[R^L{}_i \times R^L{}_j]$, there should be at least one nonzero element of $[R^{L+1}{}_i \times R^{L+1}{}_j]$ corresponding to the connectivity between the decomposed elements of RL_i and $R^L{}_j$.



LEGEND

$X^L = (X_i^L)$ where i ranges over all elements at level L.

X_i^L = i-th element of X at level L.

——— ASSIGNMENT (from level L) An organizational element (O^L) at a given level (L) establishes goals for the next lower level (L+1); O^L (e.g., a commander) assigns goals (G^{L+1}), processes (P^{L+1}), resources (R^{L+1}), and organizational elements (O^{L+1}) (e.g., subordinates) to meet these goals.

- - - - ASSIGNMENT (from level L-1) O^L has been assigned goals (G^L), processes (P^L), and resources (R^L) from organizational elements at the next higher level (O^{L-1}).

——— RESPONSIBILITY Exercise of responsibility within a level for meeting assigned goals.

R-2307D

Figure 2-1. Recursive Nature of Decomposition

In addition, for each nonzero $[R^{L}_{i} \times R^{L}_{j}]$, there should be a corresponding specification of the nature of the interconnection (e.g., telephone, microwave), the attributes of the interconnection (e.g., throughput), and a specification of the type of information transmitted. Of course, the higher one is in the decomposition (smaller L), the more general and all encompassing are the information flow descriptors.*

It is also possible to include input-output connectivity in process description. This serves to express what is required for process input and output, as opposed to what is actually available, and any inconsistency between it and the resource connectivity matrix can serve as an "error signal" and stimulus to an organizational element for resource reassignment. In a manner analogous to the resource connectivity matrix, we define at level L the information flow requirements matrix, or process connectivity matrix, $[P^{L}_{i} \times P^{L}_{j}]$;

$$[P^{L}_{i} \times P^{L}_{j}] = \begin{cases} 1 & \text{if resource } P^{L}_{i} \text{ requires information from process } P^{L}_{j} \\ 0 & \text{otherwise} \end{cases}$$

Again, supplementary data about the information required for each nonzero element of P should be provided.

Finally, in a similar vein, we can represent coordination among organizational elements at the same level via the coordination matrix at level k, OC_{k}:

$$[O^{L}_{i} \times O^{L}_{j}] = \begin{cases} 1 & \text{if resource } O^{L}_{i} \text{ coordinates with } O^{L}_{j} \\ 0 & \text{otherwise} \end{cases}$$

where, again, supplementary data about the nature of the coordination must also be given.

Note that since the dual of any square matrix is a graph, the foregoing matrices can be used directly to generate resources connectivity trees, process information flow diagrams, and organization charts.

2.3.6 Cross-Referencing and Redundancy Requirements: Assignment and Assignability Matrices

If each of the four dimensions were decomposed separately, it would be virtually impossible to maintain consistency across the dimensions at any

*Note that in the manner analogous to "indirect addressing" in computer systems, the location where supplemental information data are stored could be used instead of a "1" in $[R^{L}_{i} \times R^{L}_{j}]$.

given level of detail. However, tying the decompositions together provides the basis for a consistent, balanced, and cross-referenced system description methodology. This was a major requirement for IAT development.

Thus, at any level L in the decomposition, it is necessary that:

1. A process description contains references to the resources required for its performance, as well as references to the organizational element responsible for monitoring and/or controlling the process and to the goal (i.e., performance requirement) that the process must meet. Finally, it contains references to the input and output functional dependencies between itself and those other processes at the same level with which it is directly related.

2. A resource description contains references to the process(es) which that resource is assigned to support, as well as references to the organizational element responsible for the resource. It also contains references to the physical connectivities between itself and those other resources at the same level required to support a specific process.

3. The description of an organizational element contains references to the processes that the element is responsible for monitoring and/or controlling, as well as references to the resources which are assigned to that element and to the goal(s) for which it is responsible. It also contains references to the lines of authority, responsibility and coordination between itself and those other organizational elements both at the same and other levels as required to attain a specific goal.

4. A goal description contains references to the organizational element responsible for the attainment of that goal as well as the process(es) for which that goal is a performance requirement. It also contains references to the higher-level goals of which it is a part, as well as to the lower-level goals which must be met in the interests of its own attainment.

This cross-referencing is accomplished by means of assignment matrices. Thus, at any level L, any of the four dimensions can be described by a composite vector (matrix) of four parts:

$$\left[P^L \times R^L \times O^L \times G^L\right] \quad ,$$

with its primary decomposition (e.g., P^L, as in Fig. 2-2) and three cross-references (e.g., R^L, O^L and G^L). Once again, the redundancy inherent in the cross-references over four dimensions can serve as a check on consistency of the data that define the C^3 system as well as a means for detecting errors

in specifications. Another reason for such cross-referening is to force, to the extent possible, a logical consistency and balance among the four dimensions. Cross-referencing can help to insure that the four decompositions proceed in parallel, as opposed to reaching extreme depth in only one or two.

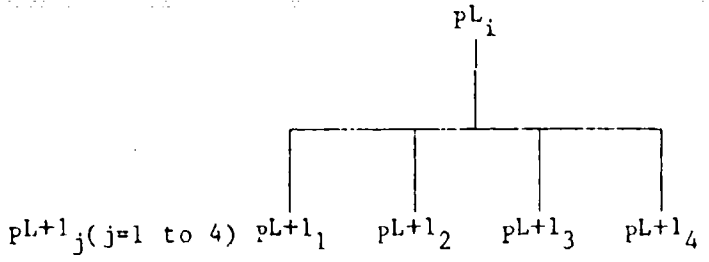


Figure 2-2. Example of Primary Process Decomposition

The various relationships described above can conveniently be represented in matrix notation as shown in Table 2-1, which presents a three-letter mnemonic descriptor, a brief definition, and a symbolic representation for each relationship.

It is worthwhile to examine in somewhat more detail the nature of the specific assignment matrices. The matrices [O×G] and [O×P] at level L, which effectively assign responsibility for goals and processes to organizational elements, are the subject of original system design and/or long-range planning in any C^3 system. On a somewhat shorter time frame, the matrix [O×R] at level L reflects the issues of resource responsibility (sometimes referred to as "ownership") and is a major focus of system reorganization and reconstitution in battle. This depends heavily upon the concept of assignability, which will be defined next.

Note that decomposition along any dimension can also be represented in matrix form. For example, the matrix $[O^L \times O^{L+1}]$ represents the organizational decomposition (i.e., organization chart) or lines of authority in the system while the matrix $[O^L_i \times O^L_j]$ represents the chart of human coordination in the system within a decomposition level. Similarly, the matrix $[P^L \times P^{L+1}]$ represents the process decomposition in the system while the matrix $[P^L_i \times P^L_j]$ represents the coordination or connectivity among processes within a decomposition level.

Finally, in order adequately to represent the inherent adaptability resulting from the capability for reassignment of resources, goals, processes and even organizational elements in a C^3 system, we must define a set of assignability matrices $[.]^*$. For example, the matrix $[R \times P]^*$ at level L must show which resources are capable of supporting (i.e., are assignable to) each process. Assignability matrices $[O \times G]^*$, and $[O \times R]^*$ can also be used to represent the capability of various organizational elements at a given level to

TABLE 2-1. RELATIONSHIPS AMONG THE FOUR DIMENSIONS

DESCRIPTOR	DEFINITION	REPRESENTATION: RESOURCE	PROCESS	ORG'L EL'T	GOAL
ISA	Is known as	(name)	(name)	(name)	(name)
AKA	Also known as	(name)	(name)	(name)	(name)
POF	Parts of	$R^{L}_{i} \in R^{L-1}_{j}$	$P^{L}_{i} \in P^{L-1}_{j}$	$O^{L}_{i} \in O^{L-1}_{j}$	$G^{L}_{i} \in G^{L-1}_{j}$
COF	Consists of	$R^{L}_{i} = \{R^{L+1}_{j}\}$	$P^{L}_{i} = \{P^{L+1}_{j}\}$	$O^{L}_{i} = \{O^{L+1}_{j}\}$	$G^{L}_{i} = \{G^{L+1}_{j}\}$
STO	Sends to; connects to; informs; coordinates with	$[R^{L}_{i} \times R^{L}_{j}]$	$[P^{L}_{i} \times P^{L}_{j}]$	$[O^{L}_{i} \times O^{L}_{j}]$	-------
RFM	Receives from; is connected to; is informed by; is coordinated with	$[R^{L}_{i} \times R^{L}_{j}]$	$[P^{L}_{j} \times P^{L}_{i}]$	$[O^{L}_{i} \times O^{L}_{i}]$	-------
ATO	Assigned to	$[R^{L}_{i} \times P^{L}_{j}]$	$[P^{L}_{i} \times O^{L}_{j}]$	$[O^{L-1}_{i} \times O^{L}_{j}]$	$[G^{L}_{i} \times P^{L}_{j}]$
		$[R^{L}_{i} \times O^{L}_{j}]$	-------	-------	$[G^{L}_{i} \times O^{L}_{j}]$
AST	Assignable to	$[R^{L}_{i} \times O^{L}_{j}]^{*}$	$[P^{L}_{i} \times O^{L}_{j}]^{*}$	$[O^{L-1}_{i} \times O^{L}_{j}]^{*}$	$[G^{L}_{i} \times P^{L}_{j}]^{*}$
		$[R^{L}_{i} \times O^{L}_{j}]^{*}$	-------	-------	$[G^{L}_{i} \times O^{L}_{j}]^{*}$

Notes:

1. Superscript = level of decomposition
2. Subscript = index
3. Read R^{L}_{i} as "i-th resource at level L"
4. Read [A×B] as A "sends to, etc; receives from, etc; or is assigned to" B
5. Read $[A \times B]^{*}$ as "A is assignable to B"

take responsibility for various goals, processes, and resources at the same level. An assignability matrix ultimately should contain as its elements (cells) an assignability index, i.e., a numerical quantity representing the relative capability of a given resource (including humans) to support a given process; or the relative capability of a given organizational element to take responsibility for a given process or resource (e.g., as a function of training, experience, or workload).

It is important to note that an assignment matrix (e.g., [R×P]) differs completely from its related assignability matrix (e.g., $[R \times P]^*$) in that it is much more sparse; of the many things a given resource could do, it is only assigned to do a small subset (perhaps only one) of them. If all resources are "dedicated" to single processes, then a square, diagonal assignment matrix results and the resource is subject only to queuing delays due to workload. On the other hand, if a given resource is assigned to support two or more processes (as is typical with both humans and computers), it is then a shared resource and is subject to contention, as well as queuing delays.

> Note that if we assume a fixed organizational structure with fixed goals, as would be typical of a "mature" C^3 system, then the actual on-line modification of such resource assignments in [O×R] and/or [R×P] within the constraints of $[O \times R]^*$ and $[R \times P]^*$ constitutes the system's adaptive capability vis-a-vis attaining its goals with its available resources. A more flexible arrangement would permit on-line reassignment of goals among organizational elements [O×G] within the constraints of $[O \times G]^*$, as an additional adaptive capability.

The requirements for combining process, resource, organizational, and goal decomposition as described in the preceding subsections provides a far more powerful tool than, for example, an $IDEF_0$ description, which at best is capable only of process decomposition with an attached indication of associated resource and "control" requirements, and no capability for representing either actual resource connectivity or organizational authority, responsibility, or coordination.

2.3.7 Depth of Decomposition

A major issue in any decomposition methodology is how to decide where and when to stop decomposing. While gross allocations of resources (including humans) among processes can often be made at higher decomposition levels using simple connectivity and aggregate performance data, the decomposition generally should be carried out to one level below that at which the user seeks the answers to specific questions. The decomposition ends at that level, with careful attention paid to: (1) the interactions among the process performance models (based on the assigned resources) representing the descriptions at this level and (2) the model parameter data, as opposed to carrying out any further decomposition. We shall return to this point later when describing the selected modeling technique.

2.4 DATA STRUCTURES FOR IAT

While the matrix notation described above assists in organizing one's thinking about the relationships among the descriptive data about a C^3 system, we also need a means of organizing the data itself to capture these relationships. Nearly any flexible, well-structured database management approach can provide such a capability. However, the use of frame/slot notation, a technique borrowed from artificial intelligence, is particularly advantageous.

Frames were originally proposed by M. Minsky as data structures for representing knowledge of stereotyped situations, such as being in well-known environments (e.g., one's living room, office, control room facility) or going to special events. Frames served as components of a broader theory of human memory and performance in their role as "units of recall," in Minsky's original conception: frames were selected from memory whenever one encountered a new situation, or made substantial changes to viewing current conditions or problems at hand. It is in this sense that frames received more widespread application as templates, i.e., "... remembered framework(s) to be adpated to fit reality by changing details as necessary" (Minsky, 1975).

Although Minsky intended frames to be employed in conjunction with associated types of information (viz., how to use a particular frame, expectations about what will happen next, what to do if these expectations are not confirmed), the use of frames within artificial intelligence (AI) modeling has focused more on the internal structural properties of frames as aids for organizing data (Schank and Abelson, 1977; Barr and Feigenbaum, 1981).

To be viewed as data structures, frames can be conceptualized as networks of nodes and relations. (Note the correspondence of this concept with two of the C^3 descriptive dimensions: resources and processes.) The top levels of a frame, in this context, are fixed, and represent conditions that are assumed to be true about a specific situation. The lower levels have terminal nodes, called slots, which are syntactically "place-holders" -- slots must be filled by particular instances or data values. Each slot can be used to specify conditions that its assignments must meet. Simple conditions are specified by markers that might require a terminal assignment to be a person, an object of sufficient values, or a pointer to data associated with another slot or another frame. More complex conditions can specify relations among data assigned to several slots (Minsky, 1975). Collections of semantically-related frames can be linked together into "frame-systems." Different frames of such systems share the same terminals: this is the critical point that makes it possible to coordinate information gathered from different viewpoints. Differences between the frames of a system can thus be used to represent actions, cause-effect relations, or changes in vantage points (e.g., from which the same data are perceived or processed).

Another important aspect of frames lies in the default values of slots. Default assignments can be used to express prototypical, potential, or acceptable values. Hence, through default values, frames can be used to represent generic information, expectations, and the like (Yager, 1984).

Default values have further uses within frame-systems. In particular, prototypical or "normally expected" values can be used to test the validity of a frame, in cases where some slots are filled in with acceptable values at the same time other slots have not (yet) been given assignments. Slots already filled-in can be used to predict the values of slots whose assignments are lacking. This feature of frame/slot notation makes these data structures especially helpful to analysts who must collect field data known to be incomplete.

For application to IAT, frames constitute major data sets for each of the four dimensions: GOALS, ORGANIZATIONS, PROCESSES, RESOURCES. "Slots" describe the data elements of a frame.

The advantages of using frames and slots are as follows:

1. Relationships that might be captured in several different matrices can be grouped together on a single frame.

2. Slots can be used with and without entries to indicate whether performance data are or are not available.

3. Default values can be defined in slots for carrying out sensitivity analyses and "zero-order" estimates of performance.

4. Cross-referencing can be handled by supplying pointers from frame-to-frame (indicated as entries on slots).

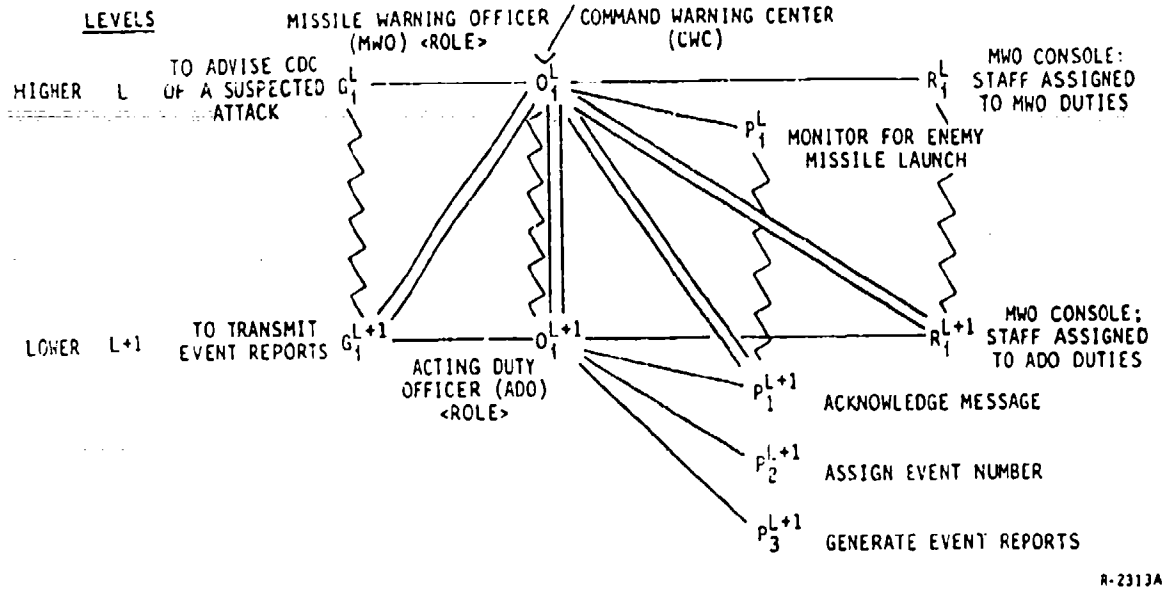
5. Slots can be added or deleted to specify attributes of the dynamic characteristics of a process.

6. The inherent nesting properties of frame and slot notation make it possible to capture information from structural models (recursive decompositions and matrices in IAT).

Figure 2-3 illustrates frame and slot notation.

2.5 C^3 SYSTEM MEASURES

We now turn to a major problem faced by C^3 systems analysts and engineers in the past. There has been a serious divergence of opinion between military operations personnel and systems designers and analysts regarding how to measure the utility of these systems. Military personnel lean toward measuring physical quantities (or their equivalent in computer simulations) which directly affect a battle outcome, such as "bombs on target," assigned targets destroyed," "attrition rates," etc. Engineers, on the other hand, tend to think in terms of the capabilities of the subsystems and systems they are designing or analyzing. As a result, some confusion has developed with regard to the meaning of various types of measures.



PROCESS NAME: MONITOR FOR ENEMY MISSILE LAUNCH

GOAL: TO ADVISE CDC OF A SUSPECTED ATTACK

ORGANIZATIONAL ELEMENT: MISSILE WARNING OFFICER (MWO) ... primary responsibility
ACTING DUTY OFFICER (ADO) ... delegated responsibility

PARENT PROCESS: ϕ

SUB-PROCESSES REQUIRED:
1) ACKNOWLEDGE MESSAGE
2) ASSIGN EVENT NUMBER
3) GENERATE EVENT REPORTS

INPUTS REQUIRED: MESSAGES - (4 TYPES)

1) INTELLIGENCE
2) SYSTEM STATUS
3) ADS
4) BSS

OUTPUTS: EVENT REPORTS - (3)

1) ADS1
2) ADS2
3) BSS

MEASURES OF PERFORMANCE:
TIMELINESS
ACCURACY

RESOURCES REQUIRED:
MWO CONSOLE
STAFF ASSIGNED TO MWO/ADO DUTIES

Figure 2-3. Example of IAT Structural Description and Process Frame for the Process "Monitor for Enemy Missile Launch" (Kornfeld, 1984)

To clarify the situation, we define the following six fundamental types of measures associated with C^3 systems:

1. System capability measures describe what a C^3 systems (or, more properly, what the system components or physical resources) can do (e.g., radar peak power, communication channel bandwidth or capacity, human cognitive and workload limitations, computer memory size, display resolution, and other attributes). All of these measures can be evaluated based on physical properties of hardware or human elements of a C^3 system, without reference to any military or natural environment. Capability measures are specific to C^3 resources and represent one class of independent variables, namely, the design variables of the system with which engineers can deal more or less directly.

2. Mission environment measures describe the situation which drives the C^3 system, i.e., those factors which consitute the threat to and the environment of the system (e.g., radar cross section of an aircraft; reflectivity and absorbance of the atmosphere; number of hostile emitters in a region; hardness of a target; flyout time of a missile). All of these measures can be determined from the military situation and environment in which a system is or might be deployed, without reference to any specific C^3 equipment or physical resources. Mission environment measures represent a second class of independent variables over which, however, neither military nor engineering personnel have any direct control.

3. System performance measures describe what the C^3 system itself (or, more properly, specific C^3 functions or processes) will do when driven by a specific threat and environment (detection range for a radar; access time to a computer memory; message queue lengths for a communications service; time required to assign a weapon to a newly detected target). All of these measures can be evaluated only if characteristics of both the situation and the C^3 system are known, since they depend on how the system drives and is driven by its environment. Performance measures are dependent variables and are specific to C^3 processes.

4. Military effectiveness measures describe what the combined effects of the C^3 system and the weapon systems which it controls will be when driven by a specific threat and environment (probability that a hostile missile is destroyed before it reaches its target; attrition rates; fraction of attackers successfully penetrating friendly defenses; fraction of attackers successfully engaged before reaching target; movement of Forward Edge of Battle Area). All of these measures can be evaluated only with respect to criterion levels set by military commanders on the scene. Effectiveness measures also are dependent

variables but are specific to military goals and of necessity must take into account weapon system as well as C^3 system performance. As an example, "number of weapons hitting designated targets" as a measure depends not only on weapon delivery system accuracy but also on target detection, recognition, countermeasures effectiveness, and other C^3 system performance characteristics.

5. System survivability measures describe the ability of the C^3 system to continue to perform one or a group of necessary processes or functions in the face of enemy attack on the system itself. For example, such a measure may be the probability that the surveillance subsystem will continue to provide at least 90 percent coverage of a specified spatial volume during the time period of the attack, with less than 5 percent down time. Similar measures may be defined for other subsystems such as communications, planning, and weapon direction and generally will be stated in terms of probabilities, times, spatial boundaries, etc. Survivability will be determined by such things as: (1) communication, or connectivity among resources in the system; (2) redundancy of dispersed resources (i.e., number of "copies" of individual resources physically located in different geographical areas); and (3) the reassignability of resources to perform different processes as was discussed in detail in subsection 2.3.6).

6. System efficiency measures describe the way in which available C^3 system resources are utilized. A highly efficient system requires a minimum of resources to do its job; thus, resource utilization in such a system should be very high. However, it is important to recognize that one cannot have both high efficiency and high survivability at the same time in the same system. Since, as noted in item (5) above, survivability is achieved via geographically dispersed multiple copies (i.e., redundancy) of resources, then resource utilization and hence efficiency must necessarily be low in a highly survivable system.

Note that it may not be feasible to improve a system design with respect to effectiveness, efficiency and survivability simultaneously. However, given the development of appropriate mathematical tools, it should be possible to modify a system design in order to:

- Improve effectiveness subject to minimum constraints on survivability and efficiency; or
- Improve efficiency subject to minimum constraints on survivability and effectiveness; or
- Improve survivability subject to minimum constraints on effectiveness and efficiency.

Such tools would make it possible to determine whether one or another of the constraints must be "broken" and by how much, in order to reach a given level of improvement (i.e., it may turn out that a slightly lower effectiveness can result in a major improvement in survivability for a given level of efficiency; or that a small decrease in efficiency can result in a major improvement in effectiveness, etc.). The trade-off possibilities simply cannot be ignored.

It is especially important to distinguish clearly between these types of measures when attempting to evaluate and/or predict human performance in these systems and its impact on system performance and effectiveness.

2.6 POSSIBLE APPROACHES

The definition and evaluation of C^3 system measures can be supported in three different ways (in increasing order of difficulty and expense, but in decreasing order of validity): by analysis, modeling and simulation, and experiment. Analytical techniques include PERT/CPM, queuing network theory, and related methods. These methods are primarily limited regarding the size (i.e., depth of decomposition) of the systems that they are able to represent. Modeling and simulation techniques include the use of such standard languages as SLAM, SAINT, and CSMP as well as more elemental methods such as Petri net modeling.

In this section we have reviewed the primary requirements and constraints for C^3 system description, decomposition, measurement, and data representation. Because of its generalizability and wide applicability as well as its direct capability for decomposition, we have selected an extension of the Petri net methodology for further development. In the following section we shall describe a Petri net modeling and simulation methodology that meets these requirements and at the same time allows us to define the significant system measures and extract both structural and quantitative measures.

SECTION 3

STAPNs: A FORMAL MODELING AND ANALYSIS METHOD FOR IAT

3.1 INTRODUCTION

In Section 1 we noted the differences between C^3 Systems and other large-scale systems. These differences comprise one major reason for the difficulty experienced in attempting to analyze C^3 systems. A second major reason has been the lack of a complete, well-structured intellectual framework which can capture all essential elements of C^3 processes, resources, and organizations. More useful analyses should be possible if we can (1) create a formal analysis framework for describing C^3 systems which meets the requirements set forth in Section 2; (2) use that framework to identify a precise, independent, and complete set of variables or measures with which to describe a C^3 system's behavior; (3) use the same framework to develop quantitative relations among the variables; (4) deduce numerical values for each of the variables using empirical or theoretical data; and (5) use the same framework for representing and analyzing both human and system activities.

This section describes ALPHATECH's approach to meeting these requirements. It is based largely on recent work by Tenney (1986) and Blitz et al. (1985). We first summarize the minimum set of physical constructs required for C^3 system modeling. This is followed by a short tutorial on the technical foundation (i.e., mathematical constructs) underlying the methodology, namely, Stochastic, Timed, Attributed, Petri Nets (STAPNs). The reader will note that the formal structure of STAPNs is directly analogous to the empirical structure of C^3 systems; it is this analogy which forms the basis of our approach.

We then introduce the concept of the "box node," a new Petri net primitive. Next we describe the tight relationships between the physical attributes of a C^3 system and the mathematical modeling formalism of STAPNs.

Finally, in the remainder of this section we discuss four additional topics:

- Relationship of STAPNs to existing methods of system representation, modeling and analysis
- Guidelines for constructing IAT process models with Petri nets
- How STAPNs can be used to model human operator and decision-making activities
- Which IAT questions may now be addressed and which require further IAT development

Details of the methodology are presented in Appendix A which describes (1) the canonical performance measures which are directly derived from the STAPNs; (2) the formal method of model refinement via model decomposition and enhancement; (3) relationships among the measures at different levels of decomposition; (4) available techniques for evaluating the measures; and (5) open issues, i.e., problems which have not yet been solved. Subsection 3.8 contains procedures and guidelines for generating measures from STAPNs.

3.2 PHYSICAL CONSTRUCTS FOR C^3 SYSTEM MODELING

The descriptors of C^3 system behavior must relate to reality. For the sake of measurability and, ultimately, of performance evaluation, they must be tied to physical activities in specific, concrete ways. The approach described here injects an abstract representation between reality and the measures, and that representation must be closely tied to reality if the implied measures are to be so. With reality the basis for all that follows, consider the fundamental elements of real C^3 systems that must be captured by a modeling language and its associated measures.

How can we start to formalize descriptions of C^3? To begin, we can adopt the following point of view:

> A Command and Control system, together with its environment and associated weapon systems, consists of a number of objects moving through neighboring geographical regions or flowing through various interconnected facilities, all in a partially coordinated, asynchronous fashion.

Each of the terms in this statement is made more precise below.

3.2.1 Objects

Many of the "objects" that "move through" a system are easy to identify: ships, aircraft, supplies, spare parts, and people. Other, less tangible, things can be said to move as well, notably information. For the sake of realism, the approach herein prefers to identify intangible entities, such as information, with their physical manifestations, such as messages. Nothing is lost this way -- information flows if and only if messages move -- and easily interpretable (and measureable) quantities can be more readily defined in terms of physical objects than in terms of intangibles.

The benefit of this point of view is clear. A vision of many objects moving around a set of regions or facilities immediately suggests many natural variables which describe their activities: velocities, rates of arrival and departure, time delays, and the number of objects in each region or facility. Each of these measures can be evaluated for the real system by observing the objects in action, so they are clearly measurable.

3.2.2 Geography

The objects that move through geographical regions include battle forces, civilian vehicles, civilians themselves, and supplies. In all cases, the motion can be considered in two distinct ways: as differential equations of motion (or equivalents) or as queuing processes (flows through networks).

The first, somewhat microscopic, view follows Newton and Euler: describe each object by continuous quantities such as position, orientation, mass, and velocity. Then, from knowledge of the forces acting on each object at any time t, deduce the positions and velocities at the next point in time t+∂. The principal advantage of this representation is that it can be made arbitrarily realistic. It is ideal for many purposes: design of flight control systems, deduction of ballistic trajectories, few-on-few combat modeling, and so on. However, it suffers from complexity when used in conjunction with models of C^3 systems: an extremely large number of objects (friendly, enemy, and neutral) can interact with the system, and to describe each of these objects with even a six-degree-of-freedom model would be quite unwieldy.

The second, more macroscopic, view suppresses exact positions and velocities in favor of discrete quantities such as regions in position-velocity space. Geography breaks into regions, altitude into zones, and movement into segments of a standard mission profile. Objects move from region to neighboring region, altitude to adjacent altitude, and segment to subsequent segment. The advantage of this representation is that noxious detail can be suppressed. Unfortunately, either accuracy suffers to some extent, or complexity dominates. Objects' behavior cannot be modeled arbitrarily accurately without making regions smaller, and hence more numerous.

The choice between the two alternatives clearly depends on the uses to which the representation is to be put; we claim that the second approach is appropriate for studies of C^3 systems. This assertion is taken to be axiomatic in what follows, but it can be justified on three empirical grounds.

The first argument is based on existing military views of the world. For example, an air defense environment is not characterized operationally by the detailed positions of every threat; rather threats follow one of several types of trajectory (high altitude, skimming, ballistic, or depressed) as they migrate from the outer air battle, to the inner air battle, to point defense. Ground operations are organized around regions of control. Air defense is managed by sectors. Even space defense is layered in several phases, and targets move from phase to phase. Discrete views of the world are ubiquitous in military life, so operational personnel readily relate to discrete models of their battle area. In addition, much documentation about how a system works already takes this form.

The second argument is based on pragmatism. Does the overall behavior of a C^3 system really depend on the exact locations of all objects in the battle area? If so, the system is extremely sensitive to events in the battlefield -- and likely to fail if events do not go as planned. Conversely, C^3 systems are

designed to be robust, so that their performance should be largely indifferent to replacing continuous movements with intermittent transitions between discrete regions.

The third argument is based on utility. To evaluate measures using a continuous model is usually expensive, as one must integrate large numbers of simultaneous differential equations. To evaluate measures using discrete models is much easier. Specifically, consider how to simulate continuous and discrete models of the same system. In terms of both software development and execution time, the discrete model is far more tractable. Except in critical situations where accuracy is paramount, discrete models often yield acceptable evaluation results with far less investment in software and analysis than do continuous models.

3.2.3 Facilities

Not all objects move around in an unconstrained manner. Many more flow along established pathways. Most importantly, the objects that move within a C^3 system are primarily messages, and these travel only over communications channels.

The basis of any C^3 system is its underlying communications network. "Communications network" is to be interpreted in the broadest sense: it includes both expensive, special purpose systems (e.g., NTDS, JINTACCS, JTIDS, etc.) as well as inexpensive, general purpose mechanisms (e.g., memos, telephones, face-to-face conversation). The nodes of a communications network reside in various facilities such as sensors, command centers, and weapons. Clearly facilities, or points within facilities, are discrete entities between which messages flow.

3.2.4 Connections

Geographical regions border one another, and objects can physically move only from one region to a neighbor. Facilities are interconnected by communications links, and messages can move only along communications links. Thus both types of object move in a way which is determined by the topology of the geographical cells and the communications network.

Connections between the geographical and C^3 networks are provided by sensors and weapons. If an aircraft enters the region surveilled by a radar, at some point that radar will begin to introduce detection and tracking messages into the C^3 system. Similarly, if a message arrives at a cruiser commanding it to launch a helicopter, a new object will begin to move through the sectors of a battle group.

3.2.5 Parallelism/Asynchrony

The laws which govern the movement of objects through regions or facilities are very different from usual physical laws. No force fields permeate

the entire system, affecting every object simultaneously and continuously. Instead, most objects move independently of one another: satellite motion and logistics messages have little influence on one another.

Movements are not completely independent, of course. Coordination does take place, but only at discrete points in time and space. For example, air engagements start when an interceptor, a target, and a command message allocating that interceptor to that target all exist in the same region at the same time -- and the interceptor may wait as long as necessary for the command before starting the engagement.

In general, coordination takes place in geographical spaces only when two objects occupy the same region. Coordination takes place in the C^3 system only when two pieces of information converge at the same facility. Coordination takes place between objects and messages only when an object causes a sensor to emit a message (so the object and sensor are in the same region), or when an engagement starts (so a message and weapon are at the same facility, and the weapon and target are in the same region).

Coordination takes two forms. In the first form, two activities start simultaneously. For example, an order to move to a heightened state of combat readiness causes many activites to start at once. In the second form, movement of one object halts until some other event takes place. For another example, an F-15 will stay on a prescribed patrol route until either a message arrives assigning it to investigate a radar contact, or an indication arrives that the plane is low on fuel.

Both of these coordination mechanisms are asynchronous: there is no worldwide clock by which events can be orchestrated. (Even systems which try to establish a common time reference ultimately rely on asynchronous protocols to exchange information.) An activity starts if, and only if, certain preconditions are satisfied and proceed for some period of time. The activity ends only when the time required to carry it out has passed, and conditions for the next activity to start are satisfied.

As a final example, consider a coordinated missile launch against a collection of ground targets. The missiles may be launched simultaneously -- coordinated by the transmission of a single command, and enabled by preparatory activites such as fueling and targeting. However, each missile will be affected by different sequences of events as defensive systems attempt to engage it -- the missiles move along their flight paths independently and in parallel. The defensive system reacts to the missiles' actions, and engages missiles in different areas with different sensors and weapons -- coordinating its activities using asynchronous track handovers and weapon allocations.

3.2.6 Complexity/Hierarchies

Numerous references to the complexity of real C^3 systems have already been made. The normal mechanism for coping with complexity is to view a system at several levels of detail. While this is always somewhat artificial,

the advantages of gaining a general perspective usually outweigh the attendant loss in accuracy, at least at first.

The perspective enunciated above supports multiple levels of detail. Objects can be decomposed: an entire Naval battle group may be a single object at the level of national policy making, but broken down into individual platforms for operational control. Geographical areas can be decomposed: air defense districts break into zones, which break into regions, which break into sectors. Facilities can be decomposed: an Army divisional headquarters is a single facility when viewed from the theater level, but an entire network of facilities when described in a procedures manual. As previously described in Section 2, successive decomposition from aggregated overviews to refined details provides the key to manage C^3 system complexities.

3.3 MATHEMATICAL CONSTRUCTS FOR C^3 SYSTEM MODELING

Turning from the physical reality of C^3 systems to mathematical abstractions, recall that we seek a representation which captures all of the characteristics discussed in Section 2 and subsection 3.2, yet which is formal (i.e., well defined). It would be a mistake to attempt to build such a representation from whole cloth, since many modeling frameworks already capture several of these features. Unfortunately, some tailoring is necessary, since no such framework deals with all aspects of C^3 in a manner which can be tied to quantitative measures.

One of the more intriguing bases for C^3 modeling is the field of Stochastic, Timed, Attributed Petri Nets (STAPNs). Invented in the early [illegible]'s (by Petri) to characterize concurrent operations in computer systems (described as networks), Petri nets have been extended over the years to capture almost all of the important aspects of large man/machine organizations (such as attributes, timing relations, and stochastic events) (Peterson, 1981). Their greatest appeal is their conceptual simplicity. Also, quite natural behavioral variables accompany each simple element of a STAPN. In addition, the topology of a STAPN model automatically determines a number of relations between these variables.

With these advantages, STAPNs are a logical choice for meeting both the descriptive and modeling requirements for C^3 systems. However, the existing STAPN literature is rather fragmented and somewhat inconsistent in conventions and terminology, so a brief (but complete) description of the version of STAPNs which is most appropriate for this work follows.

3.3.1 Tokens/Timing Models/Attributes

All Petri nets are based on a vision of tokens moving around an abstract network. Tokens are conceptual entities, meant to model the objects which move in a real network. In their simplest manifestation, presented here, tokens can be in one of two states.

When a token is created (as described below), it is always in an unavailable state. After some time elapses, the token changes to an available state. After an additional time, the token is destroyed. The interpretation of the two states is that (a) when the token is unavailable, it exists and cannot be destroyed, and (b) when it is available, it exists and will be destroyed as soon as certain other conditions are satisfied (also described below). Unavailable and available tokens will be depicted as shown in Fig. 3-1.

Figure 3-1. Token States

The time a token remains unavailable, i.e., between its creation and its entry into the available state, is determined by a timing model. Identical timing models apply to every token created by the same process -- if a series of tokens is created by one process, then a common timing model describes the length of the unavailable state for each token.

Timing models fall into four classes. The simplest models are deterministic: the duration of the unavailable state is fixed at one value, which may be zero. A more realistic model is stochastic: the duration of the unavailable state is random, but always drawn from the same probability distribution. The randomness can be used to represent either actual physical uncertainty or known modeling imprecision. In the third case, times may depend on information carried by the tokens themselves (see below). The final class of models allows the timing to depend on external variables: the duration of the unavailable state may depend on the time of day, temperature, the Dow-Jones average, or other phenomena not explicitly represented by token flows.

Finally, tokens may carry attributes along with them. Attributes are simply numbers (or other information encoded as numbers) which accompany a token through its life. Values are assigned to the attributes of a token when it is created, and they do not change until the token is destroyed, after which they are irrelevant. Attributes may be continuous- or discrete-valued, or combinations thereof in vector form.

3.3.2 Places/Decision Rules

The abstract network through which tokens move consists of two types of elements. The first type is called a place. Tokens reside in places while they are unavailable and waiting to become available, or while they are available and waiting for conditions to arise allowing them to be destroyed.

Places are depicted as illustrated in Fig. 3-2 below. Depending on the number of input and output connections, a place can play four roles; the four cases are shown separately in Figs. 3-2a -- 3-2d.

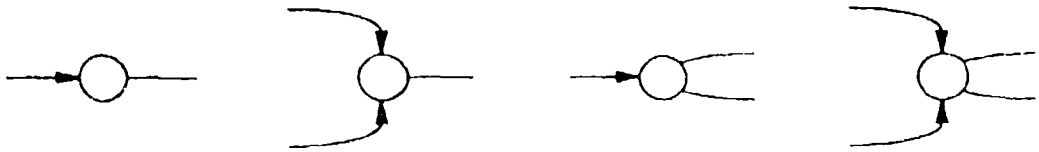


Figure 3-2. Places

The four roles are:

A. Storage: Tokens arrive in a place from one source, and depart to one destination after they have become available and the conditions for their destruction have been satisfied.

B. Confluence: Two or more streams of tokens flow together to proceed on to a single destination, after being stored.

C. Divergence: One stream of tokens is broken into two or more streams, which proceed to different destinations after storage.

D. Mixing: Two or more input streams combine, and then are broken down (usually in a different way) into streams which move to separate destinations after waiting.

In cases C and D, there must be a way to determine which path any individual token will follow out of the place. For this, we associate a decision rule with every place. By invoking the decision rule for every passing token, we can ensure that token behavior is always completely determined.

As with timing models, decision rules can vary in complexity. Deterministic decision rules, which always direct tokens to the same destination, are possible but not particularly interesting. Stochastic decision rules capture random events and cover modeling uncertainty. More complex decision rules take into account the availability of tokens at nearby places (so tokens are sent only to destinations that are ready for them) or the values of attributes attached to the token being handled. Finally, the most complex decision rules depend on external parameters, such as time.

Note that many tokens may pass through a single place. In fact, several tokens may occupy a place simultaneously. Figure 3-3 exemplifies how tokens may flow through one place:

Figure 3-3. Tokens Moving Through Places

In this figure, the first token is created and inserted into the place (Fig. 3-3a). It waits there until it becomes available (Fig. 3-3b), but may not leave the place immediately as the other conditions for destruction may not be satisfied. In fact, while the first token is waiting, a second token may be created in the place (Fig. 3-3c). Eventually, the first token will leave (Fig. 3-3d), but the second cannot leave until it becomes available.

3.3.3 Transitions/Firing Rules/Attribute Maps

The second type of element of which the abstract network is constructed is called a transition. Transitions determine how and when tokens are destroyed and created. Transitions evolve through three states: potentially enabled, enabled and disabled. Normally every transition is disabled; when certain conditions hold, a transition will become potentially enabled; if other conditions hold, it becomes enabled. In either case, the transition leaves the disabled state only for infinitesimal periods of time. After becoming enabled, the transition destroys some tokens, creates some others, assigns attributes to the new tokens, and immediately reverts to the disabled state.

Transitions are connected to upstream places, from which they take (destroy) tokens, and downstream places, into which they insert (create) tokens. The states (unavailable or available) of any tokens in the upsteam places determine when a transition becomes potentially enabled. Figure 3-4 shows how to depict a transition with one input and one output place.

Figure 3-4. Transition

Every transition's behavior is specified by a standard Timed Petri net firing rule. This rule has several parts. First, whenever at least one available token occupies each upstream place, the transition is potentially enabled. There may be several potentially enabled transitions at any one time; we rely on the decision rules at the places to select exactly one

transition to actually become enabled. Once enabled, the transition fires. Firing a transition involves removing exactly one available token from each upstream place and inserting exactly one unavailable token in each downstream place.

For example, a simple sequence of events which can take place in the model of Fig. 3-4 is shown in Fig. 3-5.

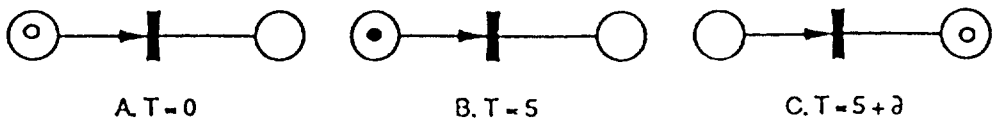


Figure 3-5. Tokens Moving at Transitions

Here, one unavailable token occupies the upstream place at time 0 (Fig. 3-5a). At some point, say at time 5, that token becomes available (Fig. 3-5b). The transition is now potentially enabled; since it is the only transition in the model, it must be selected to become enabled. Thus at time 5, the transition fires; it removes the available token from the upstream place, and deposits a new, unavailable token in the downstream place (Fig. 3-5c).

What if the transition has more than one input or output place? This slightly more complex situation appears below in Fig. 3-6. In this example, the system starts at time 0 with an unavailable token in P_1 (Fig. 3-6a). Again at time 5, this token becomes available (Fig. 3-6b). However, T_1 is still disabled, since no token occupies P_2. Even if another token were to arrive and become available in P_1, T_1 could not fire (Fig. 3-6c). Also, the arrival of a new token in P_2 at time 7 (Fig. 3-6d) is not sufficent to enable T_1, until that token also becomes available, say at time 18 (Fig. 3-6e). At this point, T_1 becomes enabled; it fires and removes one available token from each of P_1 and P_2, and places unavailable tokens in P_3, P_4, and P_5.

Figure 3-5 showed how transitions act as (a) separators between places, (b) destroyers of available tokens, and (c) creators of unavailable tokens. Figure 3-6 exemplifies the fourth crucial job of transitions: to coordinate the flow of tokens. Any available token in P_1 must wait for another available token to arrive in P_2 (and vice versa). The apparent complexity of the firing rule comes from a need to deal with special situations. In these, two or more places are potentially enabled simultaneously, but are related in such a way that if any one fires, then they all become disabled. Transitions in this situation are said to be in conflict. A simple example of a conflict situation is presented in Fig. 3-7. Initially, all transitions are disabled (Fig. 3-7a). At some later time, a token is created in P_3 and, later, becomes available (Fig. 3-7b). Now, both T_1 and T_2 are potentially enabled. If T_1 fires, it will remove tokens from P_1 and P_3, so T_2 is disabled along with T_1 (Fig. 3-7c) since no token remains in P_3. If T_2 fires, a symmetric situation arises (Fig. 3-7d). With no other information, the behavior of the model is

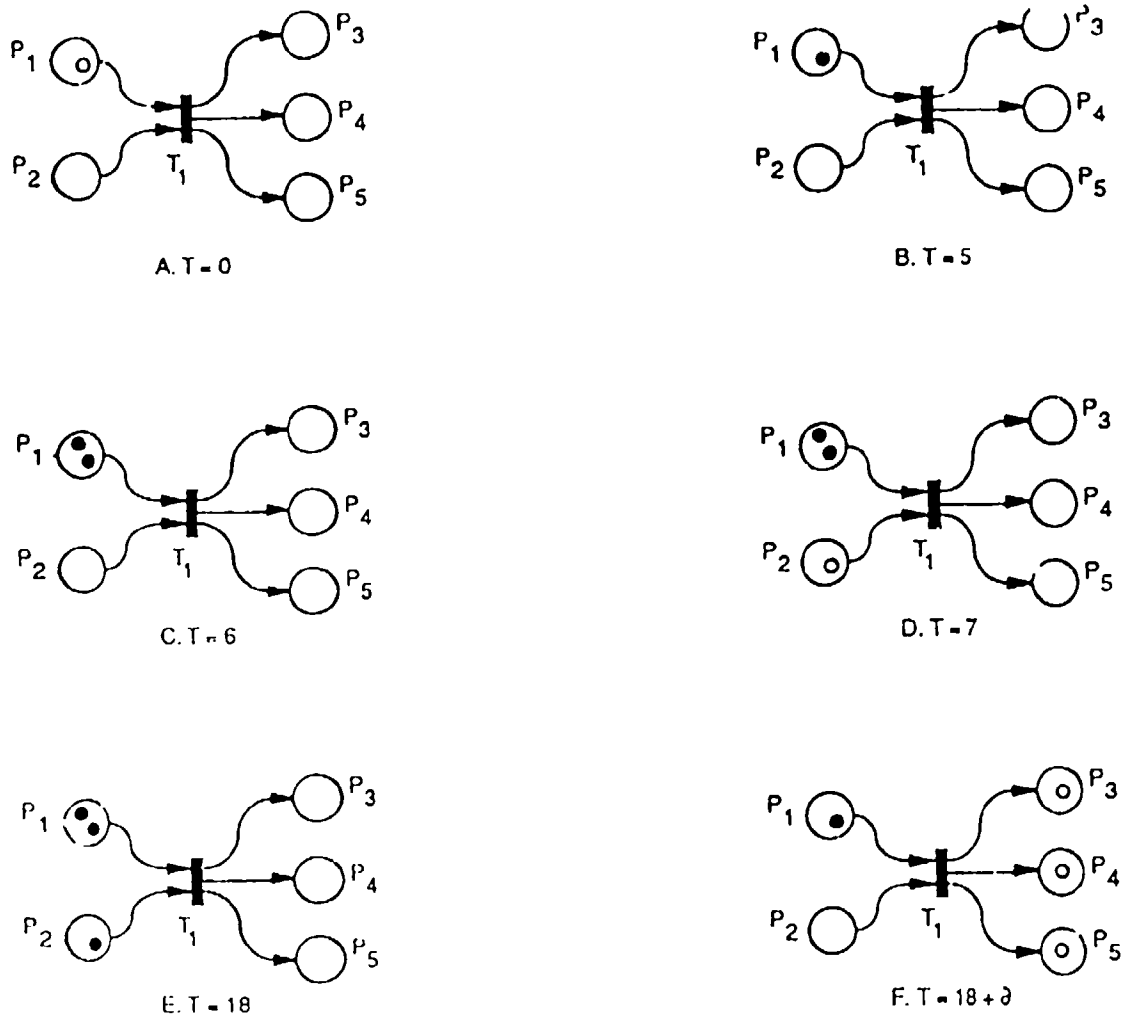


Figure 3-6. Coordination by a Transition

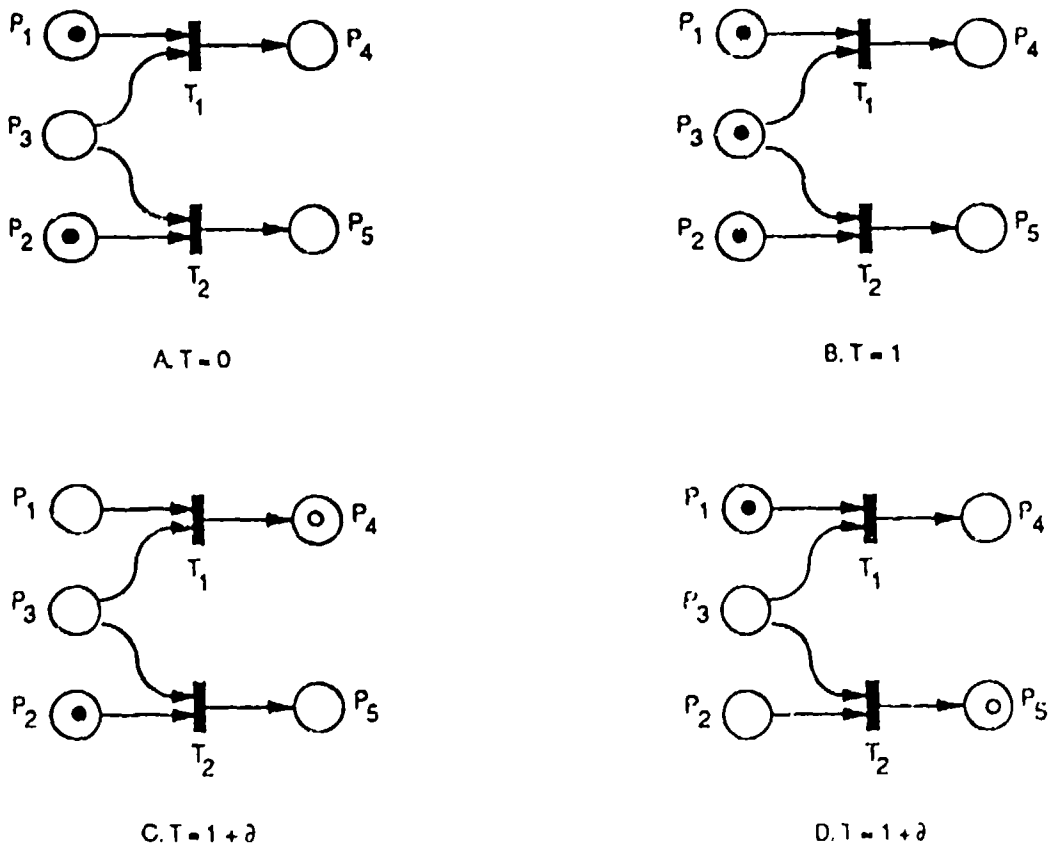


Figure 3-7. Transitions in Conflict

not well defined. However, the decision rule at P_3 specifies the transition to which the token in P_3 is allocated, and so resolves the ambiguity in a well defined manner.

The final aspect of a transition is the mechanism which determines the values of attributes attached to newly created tokens, called an attribute map. Like timing models and decision rules, attribute maps may be deterministic or stochastic, or may be dependent on external parameters. The most useful form of attribute maps deduces attribute values for newly created tokens from attribute values carried by tokens destroyed in the same firing.

3.3.4 Arcs/Petri Nets

The connectors between places and transitions are called arcs. Only one basic constraint restricts the placement of arcs in a STAPN: arcs can only connect places to transitions, or vice versa; they can never connect places to places or transitions to transitions. In addition, we impose two additional constraints, not absolutely necessary to Petri net theory, but which make for more meaningful measures.

First, every place must have at least one arc leaving it. This precludes the obvious degenerate situation where tokens pile up in a place ad infinitum. (There may be less obvious cases where this occurs due to some pathology of the Petri net, but these cannot be ruled out by simple topological constraints.)

Second, multiple arcs between one place and one transition are prohibited. Often these are used to create or destroy several tokens in one place simultaneously. However, the use of multiple arcs significantly complicates the derivation of relations between measures in two or more models. Moreover, activities modeled by multiple arcs can be modeled by equivalent nets with single arcs, albeit with a slight increase on the model's complexity.

A collection of places, transitions, and arcs which satisfies these constraints forms a Petri net. With the addition of timing models, decision rules, and attribute maps, the Petri net becomes a STAPN. With one other piece of information, the STAPN becomes a complete model of a system.

The additional information is the initial marking of the net. A marking completely describes the state of the STAPN. It includes the number of tokens in each place, whether each token is available or unavailable, and the attribute values attached to each token. It also includes, for every unavailable token, the amount of time remaining until the token becomes available (this can be considered to be an attribute carried by all tokens).

The marking of a STAPN captures all of the memory of the process which it models. It is relatively easy to show, by direct construction, that:

Fact 1: Given a STAPN, and the marking of that STAPN at any time t. If the timing models, decision rules, and attribute maps do not depend on external parameters, then the probability distribution

over the set of markings at any future time t+T is completely determined. If the timing models, decision rules, and attribute maps are deterministic, then a unique marking for any future time t+T is completely determined.

As an example of a complete STAPN, consider Fig. 3-8, which shows a network and its initial marking at time zero.

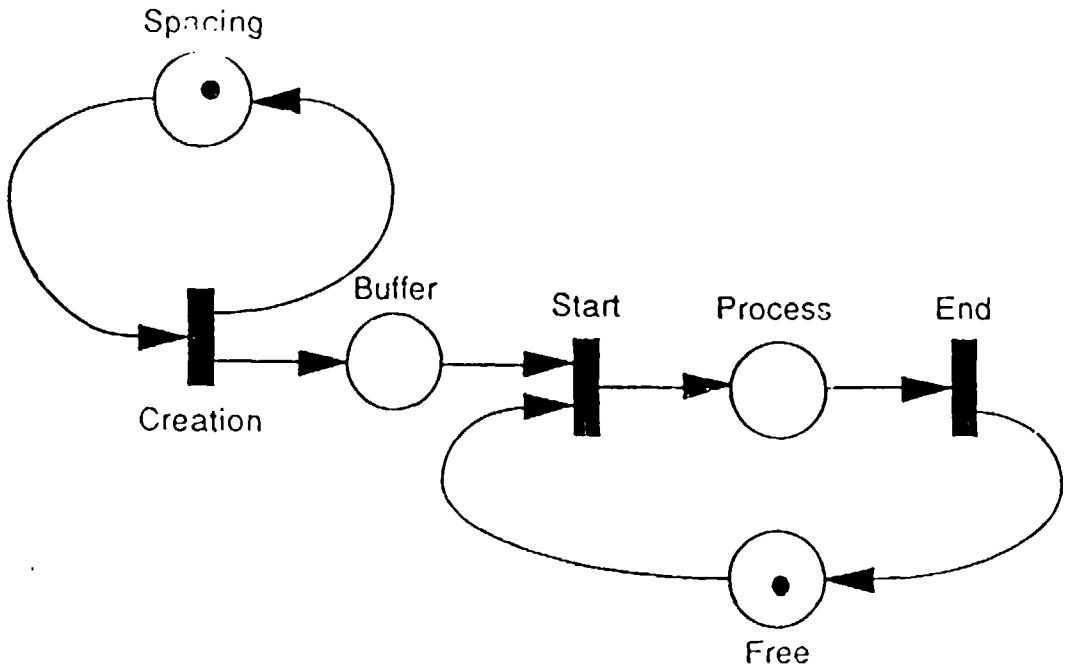


Figure 3-8. Example STAPN

This system operates as follows. The token in SPACING immediately enables CREATION, which places a token both in BUFFER and back in SPACING. If the timing model for this token is deterministically zero, then CREATION can fire immediately. More realistically, the timing will be nonzero, so the token rests in SPACING before enabling CREATION again. Once this happens, this cycle repeats: tokens are intermittently deposited in BUFFER ad infinitum.

There is another set of events that are partially coordinated with the CREATION process. These begin when the first token in BUFFER becomes available. START fires, and when the token created in PROCESS becomes available, END fires and restores a token to FREE. However, if the time to become available in PROCESS is long compared to the time in SPACING, more tokens will have arrived in BUFFER before the token in FREE becomes available. Alternatively, if the time spent in PROCESS and FREE is small compared to the time spent in SPACING, the FREE token will wait for the next arrival in BUFFER before firing START.

Overall, then, this is a model of a source of tokens which are stored in a buffer until a resource becomes free. Then, a process starts; when it ends, the resource is freed up and made available to work with another token in the buffer. (In fact, if the timing models for SPACING and PROCESS are probabilistic with exponential distributions, and all other timing models are zero, this is identical to the simplest model from queuing theory: an M/M/1 queue. More generally, any queuing theoretic model can be represented by a STAPN; the converse is not true.)

3.3.5 Complexity/Hierarchies

How can STAPNs handle this kind of complexity which $IDEF_0$ was designed to deal with? As mentioned in Section 2, the normal mechanism for managing complexity is to organize knowledge in a hierarchy. Can STAPNs be arranged in a hierarchy? As we will see in subsection 3.5, the answer is "yes" -- one can consider a large STAPN to be an interconnection of subnetworks, each of which is a STAPN itself. Each of these subnetworks may be further divided into subSTAPNs, and the process iterated until each subnetwork reduces to a single place or transition.

In addition, timing models, decision rules, and attribute maps which appear in a crude STAPN model of a system may be quite complex. Rather than leaving them as primitive entities, their descriptions may expand into entire STAPNs when more detail is added.

3.3.6 The "Box Node"

In order to manage the hierarchical complexity typical of C^3 systems and, more specifically, to be able to represent successively higher levels of aggregation of other Petri net primitives in simple graphic form, ALPHATECH has added the concept of a "box node" to standard Petri net representations.

In the formulations we have described so far, STAPN models appear as flat graphs even though the C^3 systems they are intended to represent are hierarchical in nature. Consequently, the STAPN model of a C^3 system reveals all of the details of that system without any organization in this presentation. To capture the hierarchical nature of C^3 systems, ALPHATECH created a new Petri net primitive -- the box node. Box nodes, which are represented in a Petri net diagram as rectangles with a darkened stripe in the diagrams, are used to cluster and conceal other Petri net primitives -- places, transitions, and other box nodes -- which form a subnetwork or submodel (see illustration in Fig. 3-9). A submodel concealed within a box node is incorporated as a single unit into a model under construction.

The use of box nodes is analogous to the use of subroutines in modular programming and affords the same benefits. Box nodes conceal detail and allow the system analyst to work at a level of abstraction that is higher than the level of places and transitions. Box nodes can be arbitrarily complex. Most importantly, box nodes allow the development of standardized process models

in modular form, thus permitting easy repetition within a simulation (e.g., as with modules representing multiple surveillance assets), as well as easy transfer of such modules to other simulations. Using the STAPN methodology, users of IAT may work with box nodes representing say, the entire surveillance subsystem of a C^3 system, where the representation of this subsystem may require many hundreds of primitives.

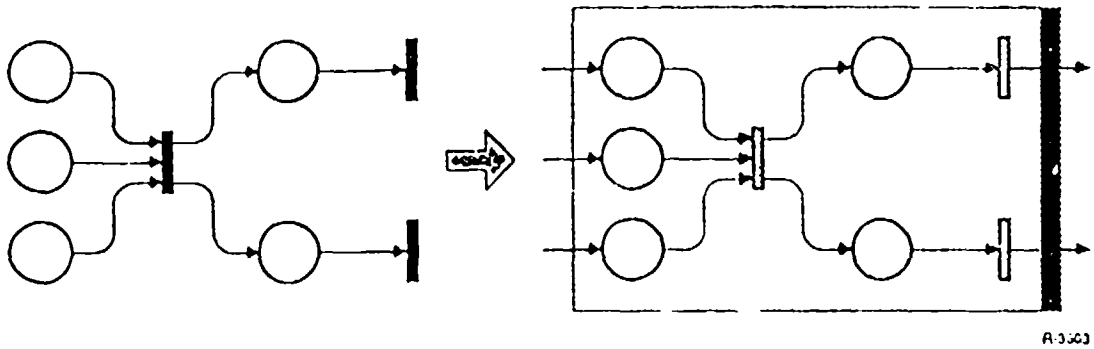


Figure 3-9. The Box Node as a New Petri Net Primitive. Within the Box Node inputs always go to places; outputs always leave from transitions.

Thus, with the addition of the box node primitive, the Petri net becomes an extremely convenient and expressive modeling device. ALPHATECH has already implemented the Box Node in its Micro Modeler for the Air Force's Foreign Technology Division (Blitz et al., 1985).

3.3.7 Implications for Precision

Because a STAPN is rigidly defined, so are all of the events that take place as it operates. Provided that the net topology obeys the interconnection constraints, and provided that the timing models, decision rules, and attribute maps are well defined for all values of the variables on which they depend, there is absolutely no room for ambiguity about how the model works (this is the essence of Fact 1). Hence, if measures are explicitly related to a STAPN's behavior, they too will be rigidly defined. Note that any model, including a STAPN, is really an analyst's hypothesis about how a system works. That hypothesis should always be subject to both reasonableness checks and validity tests on the measures taken.

3.3.8 Implications for Mutual Exclusion

The basic events of a STAPN are transition firings and assignment of tokens to places. Any measures defined for STAPNs must be related to these events. Two or more measures which are affected by the same event will be correlated, if not dependent. The criterion for mutual exclusion between measures implies a requirement that the measures be defined in terms of non-overlapping sets of these basic events.

Thus the formal structure of STAPNs provides a step towards the goal of independent, and largely uncorrelated, definitions of C system measures of performance. The requirement for mutual exclusion among the sets of events which underlie measure definitions are satisfied in Appendix A, where canonical measures are developed.

3 4 RELATIONSHIPS BETWEEN THE PHYSICAL AND MATHEMATICAL CONSTRUCTS

Earlier we alluded to certain analogies between the real C systems and the abstract mathematical constructs inherent in STAPNs. This section discusses each of these analogies and sets out the axioms of our approach.

The reader will recall the viewpoint presented in subsection 3.2:

> A Command and Control system, and its environment, consist of a number of objects moving through neighboring geographical regions or flowing through various interconnected facilities, all in a partially coordinated, asynchronous fashion.

Now consider a simple description of STAPNs:

> A STAPN describes how a number of tokens move among interconnected places in a partially coordinated, asynchronous fashion determined by transitions.

The analogies between each of the terms in these statements is made more precise below.

3.4.1 Tokens: Objects

First, messages move through a C system and objects move through its environment. Tokens move through STAPNs. In general:

> Assertion 1: All objects that move through a Command and Control system or its environment can be represented by tokens moving through a STAPN.

As used in subsection 3.2, "object" is a general notion. Objects include concrete entities such as submarines, people, and messages. Objects also include concrete manifestations of more abstract notions: the position of an indicator that represents the DEFCON state, the location of warfighting materiel that represents a state of combat readiness, or the armament on a close air support aircraft that represents its capabilities.

Unlike many objects, which exist permanently, tokens are created and destroyed regularly. For tokens to represent real objects, a succession of tokens must be used to represent an individual object, by dividing the lifetime of that object into separate segments of time. Different tokens will represent the continuous existence object during each segment. For example,

a single aircraft may fly through several geographical regions; a different token will represent that aircraft in each region. The sequence of tokens that represents the aircraft corresponds to the sequence of regions through which the aircraft travels.

As the above example shows, the correspondence between token and objects cannot be one-to-one. However, we can infer:

Conclusion 1: At any single instant of time, there should exist a one-to-one correspondence between tokens and (generalized) physical objects.

3.4.2 Places: Regions, Facilities

Objects move from region to region; messages move from facility to facility. In discrete models, movement is instantaneous and objects spend almost all of their time in regions and facilities. Tokens move from place to place, and spend almost all of their time waiting in places. Thus we claim:

Assertion 2: All facilities in a Command and Control system, or regions in its environment, can be represented by places in a STAPN.

Generally, this assertion implies that places represent processing: that the time a token stays in a place is related to the activities which involve the corresponding object in the real system. It takes time to cross a geographical region; it takes time to handle a message. Neither aircraft nor messages can proceed until certain amounts of time have elapsed. Tokens must stay in places until they become available; timing models capture the duration of this wait. Thus we have:

Conclusion 2a: Transit and processing times must be represented by timing models in a STAPN.

In reality, objects may leave one region for any one of several neighboring regions; messages may be forwarded to one of a number of facilities connected to a sender. The selection of an alternative region or facility depends on random effects, standard operating proceedures, and operational policies. Each place in a STAPN has an associated decision rule which determines which of several arcs a token will follow out of a place. We conclude that:

Conclusion 2b: Policies, procedures, and random events which influence the paths which objects follow must be modeled by decision rules in STAPNs.

Thus if a place represents an ASW sector for a Naval battle group, then a timing model will describe the time required by hostile submarines to move through the sector, and a decision rule will model the process which determines the sector into which the submarines move next.

A limitation of this approach becomes apparent when regions or facilities can change over time. If a sensor or weapon moves, then the regions which form their fields of view or fire change. If a battle group approaches a coastline, it redefines its ASW sectors to exclude areas where submarines cannot operate. In space, orbital motions cause communications links to be established and broken regularly. If a C^3 system reconfigures itself after sustaining damage, then its facilities and their interconnections change. While one can envision a STAPN model which tracks these changes, the notions required to formalize this process have not yet been developed.

3.4.3 Transitions: Boundaries, Events

Transitions demarcate and coordinate the flow of tokens. Consider the physical analogs of each of these roles separately.

Along with the correspondence between objects and tokens, there must be a correspondence between events in the real system, and the events which create and destroy tokens. Tokens endure in a place while the corresponding object occupies a certain region or facility; tokens leave a place when the object leaves. Tokens leave when a transition fires, so we must have:

> Conclusion 2c: There is a one-to-one correspondence between boundary crossings (between regions and/or facilities) by objects in a real system, and transition firings in a STAPN.

Continuing the ASW example, a submarine crosses a boundary when it enters or leaves a sector; acoustic energy crosses a boundary when it passes from the ocean to the piezoelectric crystal in a hydrophone; a message crosses a boundary when a sonar operator tells the ASW officer on duty about a new detection. In a STAPN model of this process, transition firings represent each of these events. Recall that transition firings are instantaneous; so are boundary crossings if the boundary is infinitely thin and the object crossing it always moves with nonzero speed.

Turning to the coordination role, transitions force some tokens to wait until others become available. The physical analogy to this occurs when objects or messages are prohibited from crossing a boundary until other objects or messages are in suitable locations. In general, we suggest that:

> Assertion 3: Coordination occurs in a real system only when one object or message may be forced to wait until some other object or message is ready to move on.

This is less an axiomatic statement than a particular definition of coordination. Nonetheless, it captures the essence of real coordination schemes in asynchronous systems. In fact, it even applies to synchronous systems, as certain activities cannot begin until a global clock "tick" reaches them.

Petri net transitions should be familiar to users of PERT charts: a transition provides exactly the same coordination mechanism as do PERT models.

In fact, standard PERT charts are identical to STAPNs which have (a) deterministic timing models, (b) no decision rules, (c) no attributes, and (d) no loops formed by any series of arcs. Various extensions to standard PERT models include some forms of these other features, excluding loops. Just as PERT models have been widely used in scheduling applications because of their flexibility and simplicity, so their extensions, namely STAPNs, convey the same benefits.

3.4.4 Complexity/Hierarchies

In Section 2 and subsection 3.2.6, we saw that hierarchical representations of C^3 systems were necessary to prevent their inherent complexity from overwhelming a viewer. In subsection 3.3, we saw that there is a potential for complex STAPNs to be either divided into subSTAPNs or expanded at their timing models, decision rules, or attribute maps.

If analogies, consistent with assertions 1-3, can be established between a gross model of a C^3 system and reality, as well as between a refined model and that same reality, then there must be relationships between the gross and refined models. Some general forms of these relationships are presented in Appendix A.

3.4.5 Implications for Measurability

What does it take to establish strict analogies between events in a real C^3 system, and events in a STAPN model of that system? What do we look for in the real systems to be studied, and how do we translate what we find into a STAPN model? How does all this help ensure that variables defined in terms of the STAPN are in fact measurable in the real system?

A summary of the process discussed above suggests that STAPN representations can be built systematically if we:

1. specify the objects that move through the system, and define the token attributes that are needed to characterize each,

2. specify the regions or facilities in which objects may be stored, and define a place for each,

3. specify the boundaries which objects cross as they move between regions and facilities, and define a transition for each,

4. where objects must reside in regions or facilities for a period of time before moving on, define a timing model for the corresponding transition/place pair,

5. where objects may depart a region or facility along one of multiple paths, define a decision rule for the corresponding place,

6. where attributes may change as an object crosses a boundary, define an attribute map for the corresponding transition,

7. where coordination can happen, define the transitions (and related tokens and places) which model the coordination mechanism.

If we follow these steps, the correspondence between reality and a STAPN is based on the definitions of objects, the regions or facilities in which objects may reside, and the boundaries between regions and facilities. Suppose that this correspondence is made precise for each token, place, and transition in the STAPN, and that we impose the following guideline when defining behavioral variables:

Assertion 4: All measures will be defined in terms of the STAPN elements, so that the measures relate to transition firings and the creation, storage, and destruction of tokens in places. For these measures to be completely defined, there will be an explicit way to evaluate their values by observing the behavior of the STAPN model.

Let the evaluation mechanism be captured in some algorithm, which accepts a list of firings, creations, storage, and destructions and which produces the value of a measure. Then it is reasonable that:

Conclusion 4a: Values of measures defined on a STAPN can be obtained for the corresponding real system, using the algorithm for evaluating those measures for the STAPN, by replacing (a) transition firings with boundary crossings by objects, (b) token creations by object entries into regions or facilities, (c) token storage by object residency in regions or facilities, and (d) token destruction by object departures from regions or facilities.

Moreover, since each object, region, facility, and boundary is defined in terms of physical entities, which are observable, we conclude that:

Conclusion 4b: STAPN variables consistent with Assertion 4 are measurable in the real system.

Because representations of C^3 systems are not unique, different analysts will undoubtedly choose to define objects, regions, and boundaries in slightly different ways. However, as long as they are all physically meaningful, and as long as the correspondences between real and STAPN elements are carefully defined, we can feel comfortable about defining measures for the STAPN and expect them to carry over into valid measures for the real system and a standard method for evaluating them.

The reader is referred to Appendix A for further detail.

3.5 RELATIONSHIP OF STAPNs TO EXISTING METHODS OF SYSTEM REPRESENTATION, MODELING, AND ANALYSIS

In Fig. 3-10 we show the relationships among all of the various IAT elements previously defined. Along the bottom of the figure are the various techniques used by systems analysts to gather and structure data about C^3 systems. As indicated, these can be employed, as individual familiarity with each one dictates, to obtain as much of the process and connectivity data required to develop a detailed-level C^3 system process model using the Stochastic, Timed, Attributed Petri Net (STAPN) modeling method described earlier in this Section. From the model structure itself, it is then possible to define a complete, nested set of Probability, Rate, Occupancy, and Delay ("PROD") measures for the C^3 system.

However, in order to obtain quantitative values for these measures, we need to know two things: (1) the performance capabilities of the various resources, and (2) the specific resource-to-process assignments. Of particular importance here is the data on shared resources, i.e., where one resource is time-shared among several processes. Then, to complete the picture we need scenario data to define the specifics of both enemy target and friendly weapon system actions.

Evaluation of a single target moving through the system under various circumstances can produce a PERT or critical path assessment. On the other hand, a full multi-target scenario will require more sophisticated evaluation techniques such as the application of queuing network theory or, for more complex models, the use of simulation; in either case, the full range of statistics on probabilities, rates, occupancies and delays ("PROD") will be computed. As the model becomes defined in increasing detail, additional data needs will become apparent to the analyst. Discussions of example queuing theory and PERT/CPM analyses can be found in Sections 5 and 6, respectively.

As indicated by the horizontal dashed line in Fig. 3-10, the foregoing represents the set of current IAT elements and how they relate to one another. It involves two of the four decomposition dimensions (process and resource) and four of the six types of measures (system capability, mission environment, system performance, and system effectiveness) described in Section 2.

The survivability of various C^3 functions can be evaluated in a simplified manner using the static, set-theoretic method of Wohl et al. (1981). This involves (1) defining a specific function in detail and determining the quantity and type of resources required to support that function; (2) identifying the resources actually available or assignable to carry out the function; and (3) determining the probability that the requisite number and type of resources remain available after a given (specified) attack on the C^3 system. Thus, functional survivability can be assessed in a gross, static, "before-and after" sense. However, we are not yet at the point where C^3 system survivability can truly be evaluated in detail. The main problem here is that little work has been done to define carefully the various factors affecting functional survivability other than the aforementioned gross measures. For example, time averages may not make sense because of intervening changes

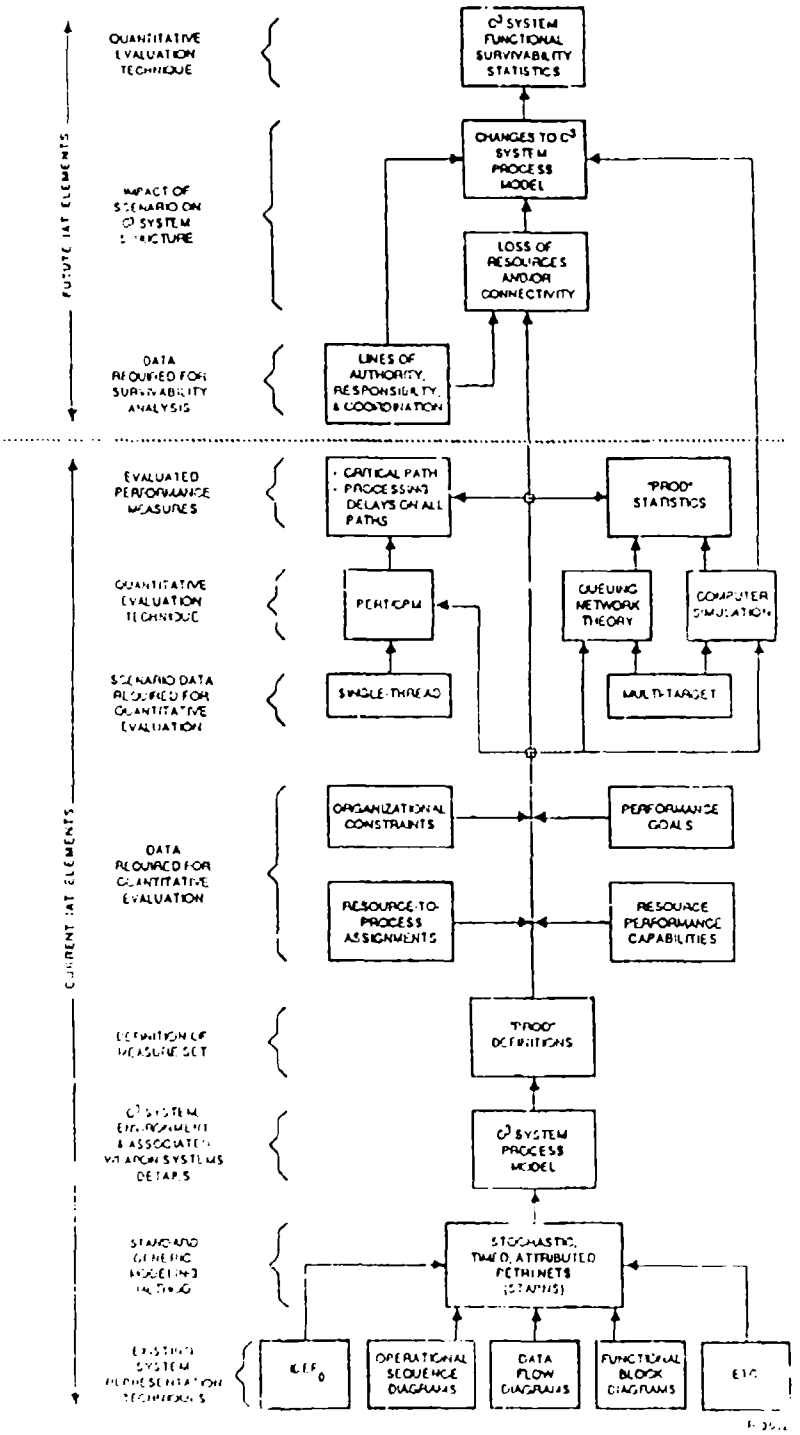


Figure 3-19. Relationship Among IAT Elements

in C^3 system topology or structure; while ensemble averages may be questionable because the system structure may change at different times in successive replications. Such evaluation would have to involve analysis of the dynamic reassignment of resource capabilities in the system, and these are determined by the organizational constraints (i.e., lines of authority, responsibility and coordination) as well as by changes in the C^3 system structure due to enemy action, etc. (See the end of subsection 2.5 for a discussion of problems associated with attempting to improve a C^3 system architecture.)

3.6 USING STAPNs TO MODEL HUMAN ACTIVITIES

In this section we examine how STAPNs can be used to represent a wide range of human activities in C^3 systems. These activities range from psychomotor tasks such as those involving both simple and complex (disjunctive) reaction times to higher-level cognitive tasks such as the Hypothesis and Option election tasks in the SHOR paradigm (Wohl, 1981).

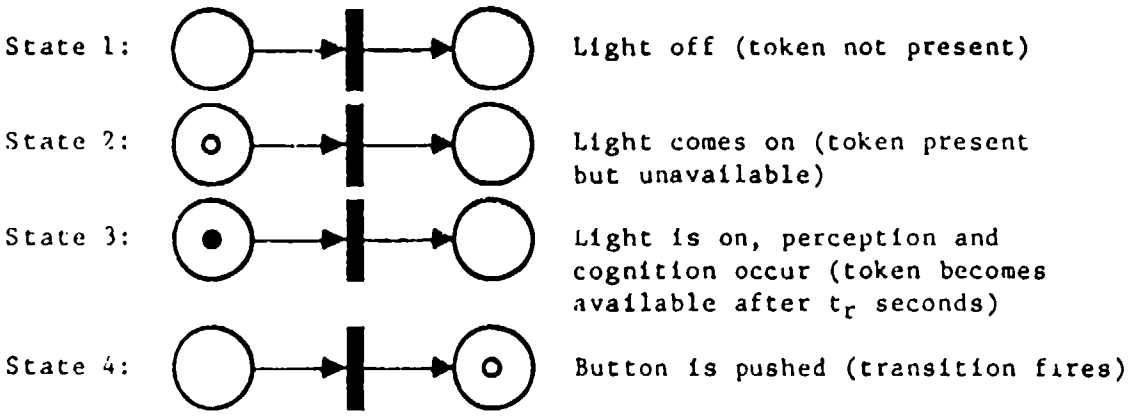
The following five illustrative cases demonstrate the applicability of STAPNs to human behavior modeling and specifically to representing the SHOR Paradigm. From them, it is clear that more complex concatenations of human activities can indeed be represented using STAPNs.

3.6.1 Case I: Simple Reaction Time Tasks

Sample Task 1: If light comes on, push buttom immediately.

Sample Task 2: If message is received, send acknowledgement immediately.

Measure: Reaction time (t_r).



Note: t_r = f(light brightness, message length, etc.)

Figure 3-11. Petri Net Representation, Case I

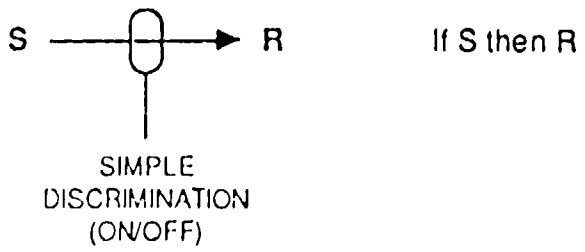


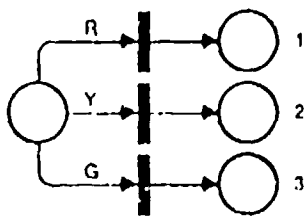
Figure 3-12. SHOR Representation, Case 1

3.6.2 Case II: Complex (Disjunctive) Reaction

Sample Task 1: There are three colored lights that can come on: Red, Yellow, Green. If Red comes on, push button #1; if yellow, #2; if Green, #3.

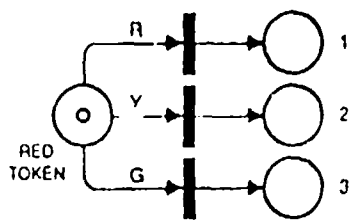
Sample Task 2: These are three possible formatted messages that can be received: R, Y, or G. If R, respond with action #1; if Y, #2; if G, #3.

State 1:



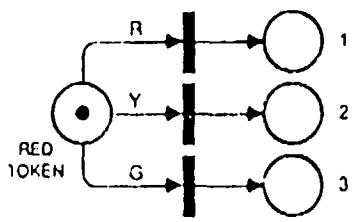
No lights on (no tokens present)

State 2:



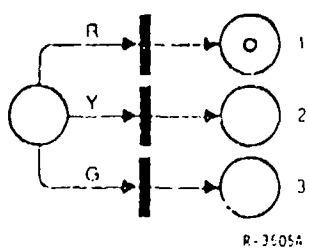
Red light comes on (Red token present but unavailable)

State 3:



Red light remains on; perception, discrimination, and cognition occur (Red token becomes available after t_c seconds)

State 4:



Button #1 is pushed (Red transition fires, Red token disappears, new token is created in place #1)

Figure 3-13. Petri Net Representation, Case II

Decision Rule: If Red token is present and available, red transition fires, etc.

Note: t_r = f(number of lights or colors, difficulty of discrimination, etc.)

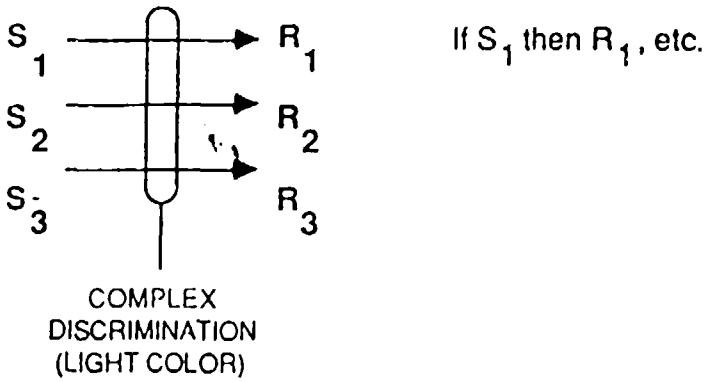


Figure 3-14. SHOR Representation, Case II

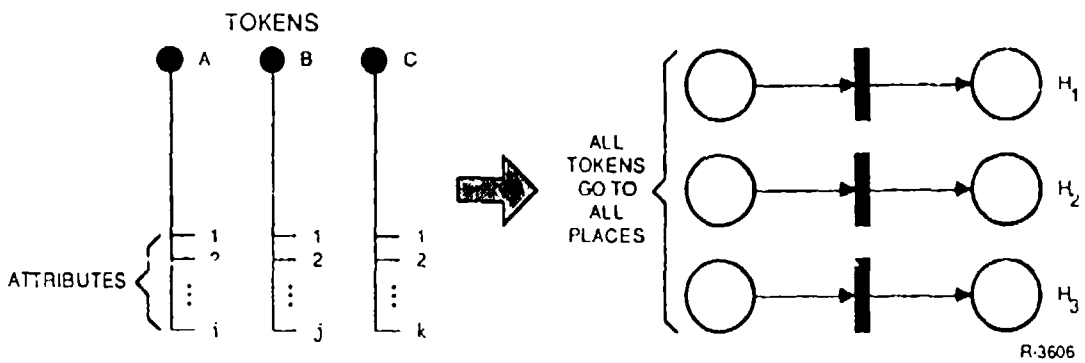
3.6.3 Case III: Hypothesis Selection Task

Sample Task: Tokens A, B, and C have attribute sets A_i, B_j, and C_k.

Token attributes represent "indicators" (e.g., diagnostic symptoms and their confidence levels).

Task: Select that hypothesis that has the greatest number of high-confidence indicators.

Measures: Processing time, confidence level in chosen hypothesis.



H_1 Decision Rule: If A_1, B_3, and C_6, then H_1

H_2 Decision Rule: If A_1, B_1, and C_2, then H_2

etc.

Figure 3-15. Petri Net Representation, Case III

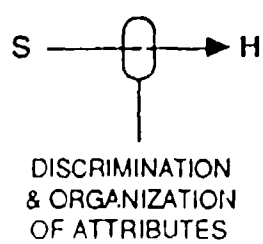


Figure 3-16. SHOR Representation, Case III

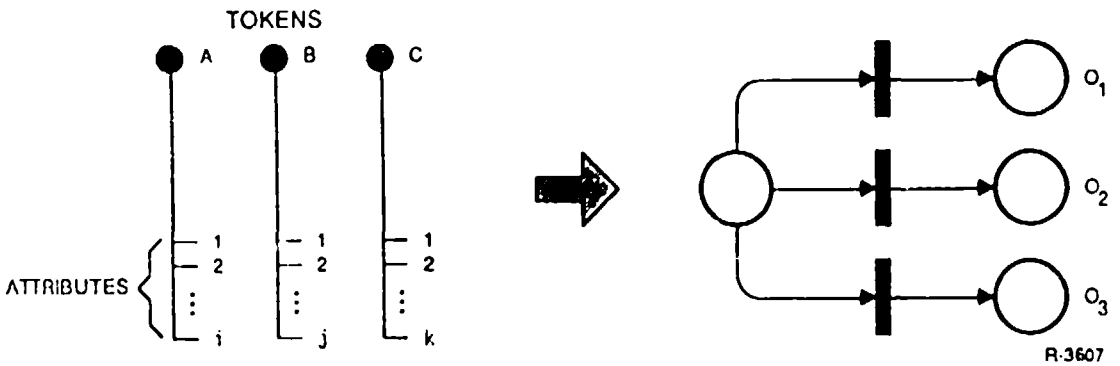
3.6.4 Case IV: Option Selection Task

Sample Task: Tokens A, B, and C have attribute sets A_i, B_j, and C_k

Token attributes represent conditions that must be fulfilled in order for an option to be viable. One condition is that a specified goal be met. Another is that a given hypothesis be correct. Subsets of the token attributes therefore serve to constrain the option set.

Task: Select that option that is least constrained and meets the goal.

Measures: Processing time.



O_1 Decision Rule: If A_1, B_3, and C_6, then O_1.

O_2 Decision Rule: If A, B_1, and C_2, then O_2 (e.g., if H_1 is correct and goal is to be met, then O_2 is least constraining option.)

etc.

Figure 3-17. Petri Net Representation, Case IV

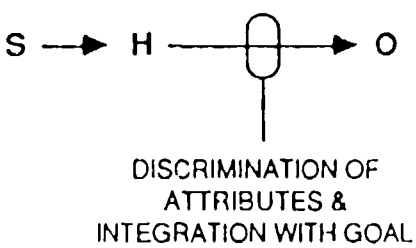


Figure 3-18. SHOR Representation, Case IV

3.6.5 Case V: Higher Level Cognitive Task: Hypothesis Generation and Testing

Sample Task: Find a rational "explanation" (hypothesis) for the presence or existence of A_i, B_j, and C_k.

Method:

(1) Look for analogies to previous situations or to other cases.

(2) Find that analogy (H_1) that best explains or accounts for the presence of A_i, B_j, and C_k.

(3) Identify additional token attributes that would have to be present if H_1 were true (e.g., C_n).

(4) Test to see whether they are present.

If so, accept H_1.

If not, try H_2 (next best analogy).

(5) etc.

Same as Case III, except for token C:

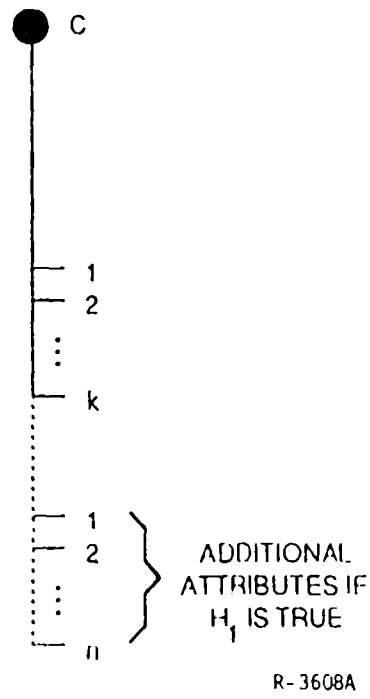


Figure 3-19. Petri Net Representation, Case V

Appendix B contains several additional examples of how various types of human interaction can be represented using STAPNs.

3.6.6 Relationship to Rasmussen's Task Taxonomy

The foregoing sample cases are also instructive from another viewpoint. Several years ago, Rasmussen proposed a simple taxonomy of human tasks which has been extremely useful in organizing thinking about human behavior in systems (Rasmussen and Rouse, 1981). In essence, he said that such behavior falls into one of three classes:

- Skill-based
- Rule-based
- Knowledge-based

Skill-based behavior is exemplified by human operators performing lower-level, more "mechanical" psychomotor tasks such as the reaction time tasks in Cases I and II above, as well as more complex tasks such as those involving tracking behavior. Rule-based behavior is exemplified by human operators and/or decisionmakers performing intermediate-level cognitive tasks such as simple pattern-matching, and hypothesis and option selection tasks in which the alternatives are known, as in Cases III and IV above. Knowledge-based behavior is exemplified by human decisionmakers performing higher-level cognitive tasks such as hypothesis and option generation tasks in which the alternatives are unknown and must be developed as part of the task, as in Case V above. This case is especially interesting since it assumes that decisionmakers are strongly dependent upon historically analogous situations as a basis for generating new hypothesis and/or option alternatives. Evidence for this assumption is contained in the work on mental models by Larkin et al (1980); on analogical reasoning by Klein and Weitzenfeld (1978); and on historical reasoning and its impact by Neustadt and May (1986).

3.7 GUIDELINES FOR CONSTRUCTING IAT PROCESS MODELS WITH STAPNs

Building IAT models of physical processes should proceed in three stages:

Stage 1: Initialization

- Scope the problem.
- Identify boundaries and data or objects crossing the boundaries.
- Set up a list of data collection goals and criteria.

Stage 2: Build a Baseline Model

- Construct a "mid-level" model of the complete system.
- Carry out data collection and assess whether Stage 1 goals and criteria have been met.

Stage 3: Refine the Stage 2 Model

- Elaborate the model structure as needed to insure consistency, completeness, and parsimony.*

Activities to be pursued at each stage are listed below.

3.7.1 Stage 1: Initialization

1. Define the boundary of the system to be modeled. Think of this in physical terms (e.g., the NCMC complex). Express the boundary in terms of data/material flows (e.g., the arriving message traffic, the departing message traffic, etc.).

2. Decompose the input/output flows into their finest elements (i.e., if two types of message might be processed differently, they form two token flows, even if they arrive through the same physical channel). Each elementary token flow defines a token type, to be placed on the list of token types, along with an annotation.

3. Define a transition for each token type to cross the boundary of the system. These transitions would be labeled "Detection message from sensor 3 arrives at NCMC" or "Operational order sent to SAC from NCMC," etc.

3.7.2 Stage 2: Build a Baseline Model

Maintain separate lists of all defined items including:

- token types
- places
- transitions
- branches

*See discussion in subsection 3.4 above.

as initialized above. Each item on a list represents a data collection goal: fill in all the information described in Section 3.4.5. Items with all slots filled in can be placed on a second list. By definition, a model is complete when all items are removed from these lists. Moreover, it is easy to guarantee consistency as each item is filled in.

Note that one may have to add items to these lists as slots are filled in. For example, if a token type "Detection message from sensor 3" is defined, that item cannot be taken off the list until two transitions, marking both the generation and disposal of these messages, are added to the transition list.

The order in which items are taken from these lists to be completed can be arbitrary. It should be selected to suit the data collector's constraints. A natural order is to walk through one processing path at a time. For example:

Q: Where do detection messages from sensor 3 go?

A: (Definition of a new place: facility, resource.)

Q: How long does it take to get there?

A: (Timing information.)

Q: What happens next?

A: (New branch or transition.)

Q: Must something else happen first?

A: (New token-type.)

etc.

3.7.3 Stage 3: Refine the Stage 2 Model

While the baseline model may be guaranteed to be complete and consistent, it may not be parsimonious. The lack of parsimony can be found in timing models and decision rules which depend on non-local state information. Additional network structure to reduce these models should be added.

3.8 PROCEDURES AND GUIDELINES FOR SYSTEMATIC GENERATION OF SETS OF MEASURES FROM STAPNs

3.8.1 Step I: Construct the Top Level Model

Section 2 presented arguments in favor of a systematic method to identify a set of variables which describe the behavior of a Command, Control, and Communications system. In this section and Appendix A the foundations for

such a method have been laid by introducing Stochastic, Timed, Attributed Petri net (STAPN) models, their canonical measures, and ways to organize such models in a hierarchy. This subsection weaves these various threads into a single, self-contained tapestry: an iterative process that simultaneously models C^3 systems and extracts a hierarchy of variables that measure their behavior.

The resulting method has five major steps, repeated until the models and measures reach a level of detail sufficent for the purpose at hand. Each subsection of this section delves into the details of each step, first in the abstract, and then in the specific context of a generic tactical air defense mission. The air defense example provides concrete illustrations of each aspect of the method. A few heuristic guidelines, generated in the course of exercising the methodology on the air defense example, bridge the abstract and concrete. These guidelines result not from theoretical analyses, but from pragmatic concerns which seem, at least in the limited context of air defense, to contribute to more acceptable and useful sets of measures.

A secondary product of the air defense example is empirical evidence that common behavior measures may apply to C^3 systems (including humans) that support a common mission area. This statement certainly applies to highly aggregate models; it may break down as the hierarchy of models becomes successively more specific. Nonetheless, a common set of high-level behavior measures does seem to exist for air defense and appears to exist as well for the other limited examples (SIMCOPE and NORAD MWC) described in Volume II.

Procedure

The first step of the methodology builds a highly aggregated STAPN model of the system under study. The purpose of this step is (a) to capture the essential objectives of the system in a STAPN framework, and (b) to establish a starting point for later iterations which refine and extend the STAPN model. The top level model need not be complicated, as the successive iterations through the methodology provide ample opportunity to add detail.

The basic process for building a STAPN model for a system was established in subsection 2.5. Repeated here, the seven substeps required to build a complete STAPN model are:

1. Specify the objects that move through the system, and define the token attributes that are needed to characterize each,

2. Specify the regions or facilities in which objects may be stored, and define a place for each type of token which can be stored in each region or facility,

3. Specify the boundaries which objects cross as they move between regions and facilities, and define a transition for each,

4. Where objects must reside in regions or facilities for a period of time before moving on, define a timing model for the corresponding transition/place,

5. Where objects may depart a region or facility along one of multiple paths, define a decision rule for the corresponding place,

6. Where attributes may change as an object crosses a boundary, define an attribute map for the corresponding transition,

7. Where coordination can happen, define the transitions (and related tokens and places) which model the coordination mechanism.

Fortunately, if the only objective is to identify behavioral measures, the final four substeps are optional. Certainly they are necessary to define a complete STAPN model. However, in subsection 2.5 it was shown that the canonical measures for a STAPN model are defined by places, transitions, and interconnections alone; the timing models, decision rules, and attribute maps contribute to evaluation of the variables, but are unnecessary for the identification of variables. Since the purpose of this work is the latter (see subsection 1.6), only the first three substeps are relevant to the general method.

Guidelines

Two general advisories apply to the construction of the top level model. First, if a model directly relates to generally accepted statements about the purpose and structure of the system which it represents, then it (and its refinements) will be more readily accepted. Since the model which represents a system is not unique (see subsection 1.5), we can always anticipate differences of opinion about the correpondence between model elements and reality, or about the appropriateness of the level of detail preserved or suppressed in the top level model. Little can be gained by offering additional opportunities for disputes about the scope or purpose of a system. Such issues have often been discussed at great length in other forums. References to published doctrine, official definitions, and generally accepted viewpoints provide a firm and easily justified starting point for the methodology.

Secondly, humans (and particularly engineers) have a great deal of difficulty describing a system at a crude level of detail. Their strong tendency is to critique a simple model of a system by pointing out numerous features which are not represented in that model. In the context of our methodology, this tendency is counterproductive: the purpose of the top level model is to capture only the broadest aspect of the system -- not every detail. More than a modicum of discipline is needed to keep the top level model simple, and to relegate the details to later iterations of the method.

In summary, two guidelines for Step I of the methodology are:

1. Maintain clear connections between the top level STAPN model and previously established views on the scope and purpose of the system.

2. Keep the top level model as simple as possible, leaving details for later refinements.

3.8.2 Step II: Generate all Canonical Measures

The STAPN model serves as an intermediary between reality and the measures sought. From the model, we can extract the set of canonical measures -- a set which is unique and well defined once the model is established.

Procedure

The second step of the methodology uses an existing STAPN model of the system under study as the basis for definitions of a large set of measures. The purpose of this step is to generate a set of measures which are (a) mutually exclusive (describe nonoverlapping sets of events) and (b) collectively exhaustive (complete with respect to the STAPN model).

The principal challenge of this step is to establish notation that is easy to extend through several levels of a hierarchy. To this end, conventions are adopted similar to those used by several other structured decomposition methods:

1. Formally name each model at level N+1 by appending a unique letter, in alphabetical order, to the name of the model at level N of which it is a refined submodel.

2. Formally name each place in a model with a letter P, using a subscript constructed as described in (4) below.

3. Formally name each transition in a model with a letter T, using a subscript constructed as described in (4) below.

4. Construct subscripts by appending a numerical index, in ascending order starting with 1, to the model name.

Canonical measures are straightforward to generate from a STAPN model of a system. The four substeps required to identify the canonical measures are:

1. To each output arc of a place P_i, leading to some transition T_j, assign a probability measure $\rho_{i,j}$.

2. To each transition T_i, assign a rate measure λ_i.

3. To each place P_i, assign an occupancy measure η_i.

4. To each place P_i, and to each output transition T_j for P_i, assign a delay measure $\tau_{i,j}$.

Note that the formal notation for each measure follows directly from the notation for places and transitions. Naturally, informal names can be attached to places, transitions, and measures for convenience. However, unless the model is to be extended to only one or two levels of detail, the formal naming method is preferable as it guarantees that each measure has a unique name.

Guidelines

Only one guideline applies to the derivation of canonical measures. Recall that tokens may have attributes. Subpopulations of tokens are implicitly defined by attributes, as we may divide tokens into groups, with tokens in each group sharing like attribute values. Often, measures may need to be defined for each attribute subpopulation separately. For example, one may wish to explicitly consider cases where the delay encountered by messages about different target types are different, since different message types may be handled with different priorities, although they follow the same processing paths. In this case, the delay for each message type, rather than an average over all messages, is of interest.

Instead of building separate models which depict the processing of each individual subpopulation, one can define vector-valued measures. The components of the vectors are measures for each specific attribute value which determines a subpopulation of interest. Thus if there are four message types, the canonical measures for the model elements which describe the message handling are four-dimensional vectors.

Fortunately, the horizontal relations defined in subsection A.3 apply, componentwise, to vectors of canonical measures as well as to scalar measures. Additional vertical relations may be required to relate components of vector measures at level N+1, where subpopulations determined by attribute values may be broken out, to measures at level N, where attribute values may not be considered at all. These relations are generally simple sums and expectations, mimicking the forms discussed in subsection A.5.

Thus the only guideline for Step II of the methodology is:

1. Use vector valued measures when the only distinction between tokens and the way they move through a STAPN is the value of attributes that they carry.

3.8.3 Step III: Select Primary Measures

While all canonical measures are physically meaningful and potentially interesting, they are not all independent. The redundant measures may be eliminated without fear of compromising the coverage of the overall set of measures, while simplifying the description of a system's behavior.

Procedure

The third step of the methodology reduces the set of candidate measures compiled in Step II using the standard horizontal relations developed in subsection A.3. The purpose of this step is simply to develop a (nonunique) set of independent primary measures.

The basic procedure passes through two phases. The first phase (substeps 1 - 4) generates a list of horizontal relations. The second phase moves through that list, successively eliminating both relations and measures:

1. Derive a conservation of probability relation between the probability measures attached to the output arcs of each place.

2. Derive a conservation of rate relation between the rate measures attached to the output transitions of each place.

3. Derive a probability-rate relation between the rate measure for each transition and the probability measure attached to every arc leading to that transition.

4. Derive an occupancy-rate-delay relation between rates at the output transitions at each place, the occupancy measure at the same place, and the delay measures between that place and its output transitions.

5. Select any relation remaining on the list.

6. Select any measure appearing in that relation.

7. Solve for the value of the selected measure in terms of the other measures appearing in the selected relation. If this is not possible, discard the relation and go back to substep 5.

8. Replace all appearances of the selected measure in all remaining horizontal relations, using the formula from step 7.

9. Discard the selected measure and relation. If any relations remain, go to substep 5.

Substeps 5 and 6, of course, are the arbitrary decision points which allow the final set of primary measures to be nonunique.

Guidelines

Several guidelines can help reduce the arbitrariness of the selection decisions. These guidelines fall into two classes: general suggestions about the measures that tend to be most acceptable to users, and specific revisions to steps 5 and 6 implied by a literal interpretation of the general guidelines.

Four general ideas about measure selection follow from considerations of the uses to which the measures will be put. First, in the evaluation of alternative systems, numerous measures quantify a system's behavior. No single measure captures a concept of good behavior; the multitude of behavioral, structural, and socioeconomic measures must be combined into a single measure of a system's worth before final selections can be made. This combination logic can simplified, and made more intuitive, if the sensitivities of the overall measure of worth, with respect to individual measures, are all of the same sign. Arbitrarily, one can ask that a selected measure has the property that an increase in its value leads to an increase in the overall worth of the system. Thus a choice between the probability of detection, and the probability of nondetection, of hostile aircraft should be resolved in favor of the probability of detection, as increases in that measure are usually considered to be good.

Secondly, the basic objective of a C^3 system is to deliver weapons to hostile targets before those targets cause any damage. The basic objective of the targets is to cause damage before the C^3 system can respond. This race between enemy assets and the C^3 processes is the essence of C^3 system evaluation, and is most naturally captured by the canonical delay measures: does the time required for an enemy to achieve its objectives exceed the time required to arrange friendly forces to meet the threat? Since the occupancy-rate-delay relations allow us to exchange occupancies and delays, they should be used to eliminate occupancies in favor of delays.

Thirdly, the probability-rate relations allow us to exclude either probabilities or rates. Except in cases where a natural benchmark (such as physical throughput capacity constraints) exist, rates can be hard to interpret. Probabilities always have such a benchmark: they are always bounded between 0 and 1. Thus probabilities should be preferred to rates, as the normalization for probability measures automatically provides a sense of scale.

Finally, division by zero is not well defined. Step 7 may result in equations involving division, and we should avoid the possibility that the denominator becomes zero whenever possible.

Thus the three general guidelines for measure selection are:

1. Eliminate measures for which decreasing values are usually interpreted as indicative of better systems.

2. Eliminate occupancies in favor of delays when considering an occupancy-rate-delay relation in substep 5.

3. Eliminate rates in favor of probabilities when considering a probability-rate relation in substep 5.

4. Do not eliminate measures which allow equations in substep 7 to have denominators which may become zero.

Mechanically applying these guidelines yields a specific strategy to generate the set of primary measures, once the horizontal relations have been identified. Noting that guideline 4 is not explicitly included, the specialized strategy for steps 5 and 6 becomes:

5a. If any Conservation of Probability relation remains, select it; otherwise, go to substep 5c.

6a. Select, for elimination, one probability measure in the relation of substep 5a for which a decrease in value is generally considered to be good. (At least one such probability measure exists, since an increase in the most favorable probability must be offset by a decrease in at least one other.) Continue to step 5c after the probability is eliminated.

5b. Select the Probability-Rate relation which contained the probability measure chosen in step 6a.

6b. Select, for elimination, the rate measure in the relation of substep 5b. (This rate measure is monotonically related to the probability measure in substep 6a, for which a decrease in the rate must also be considered to be good.) Go back to step 5a after the rate is eliminated.

5c. If any Probability-Rate relation remains, select it; otherwise, go to substep 5d.

6c. Select, for elimination, rate measure in the relation of substep 5c. Go back to step 5c after the rate is eliminated.

5d. If any Conservation of Rate relation remains, select it; otherwise, go to substep 5e.

6d. Select, for elimination, a downstream rate measure in the relation of substep 5d. Go back to step 5d after the rate is eliminated.

5e. If any Occupancy-Rate-Delay relation remains, select it; otherwise, the process is complete.

6e. Select, for elimination, the occupancy measure in the relation of substep 5e. Go back to step 5e after the occupancy is eliminated.

These substeps apply when steady state or time average measures are used; a slightly different selection order should be used for ensemble averages.

3.8.4 Step IV: Refine Primary Measures

A STAPN model of a system was built in Step I. Steps II and III developed sets of primary and secondary measures from that model. The measures are precisely defined in terms of that model. However, the measures will be used for the real system, and they may not yet be precisely defined in terms of real events.

Procedure

The fourth step of the methodology uses the list of primary measures to focus attention on some correpondences between physical elements of a real system and abstract elements of the STAPN model of that system. The elements deserving further scrutiny are simply those that appear in the definitions of the primary measures. In addition, some aspects of the measures themselves must be specified. Thus the purpose of this step is to (a) transfer the definitions of primary measures from the STAPN elements to the real system, and (b) to add further pragmatic information to the definitions. The product of this step is a glossary which augments the definitions of the primary measures.

Step IV simply revisits each of the primary measures, and the terms that appear in each, to establish more precise and unambiguous definitions; objects, regions, facilities, and boundaries modeled by a STAPN must not be ambiguously defined if a wide community of users is to share the measures produced thereby. To be evaluated consistently, the measures must be of the same type (e.g., ensemble average or steady state). The basic procedure passes through two phases. The first phase (substeps 1 - 4) generates a list of horizontal relations. The second phase moves through that list, successively eliminating both relations and measures:

Define units; define type of measure; specify T or K.

1. For each object, region, facility, or boundary mentioned in the definition of a primary measure, generate an entry in a glossary for the model which clarifies any ambiguities in that definition.

2. For each measure, specify whether it is to be evaluated as an instantaneous value, a time average, an ensemble average, or a steady state statistic. If the measure is to be a finite average, specify the interval (K or T) over which the average is to be computed.

3. For each measure, specify the units in which it is to be expressed.

All of these substeps involve the judgment of the analyst constructing the measures, so there is no objective way to evaluate the performance of this step. However, the purpose of the step is to foster communication among the users of the measures, and to insure that imperfections in this step do not propagate to later steps of the methodolcgy.

Guidelines

Two simple guidelines assist Step IV. First, the units selected in substep 3 should be as uniform as possible. There is rarely a good reason for defining some delays in days, some in minutes, and some in seconds when the overall purpose of the set of measures is considered. Translation from one set of units to another certainly does not facilitate the comparisons which the measures are to support.

Second, the definitions of measures can usually be significantly improved if they are subjected to peer review, particularly if the reviewers have widely different backgrounds. Hidden assumptions, ambiguities, and poor choice of terminology are much more apparent to someone who did not participate in the modeling process. Any differences in interpretation between reviewers can be assumed to reflect diversity present in the user community.

In summary, two guidelines for Step IV of the methodology are:

1. Use consistent units in the definition of all measures.

2. Subject the definitions and glossary to review by a set of independent reviewers of varied backgrounds.

3.8.5 Step V: Refine Model by Disaggregation or Enhancement

The STAPN model built so far may be too crude to generate measures at the level of detail required for some analysis. We must extend the model by disaggregating its elements and by enhancing it with additional structure. The refined model can then serve as a basis for an additional pass through Steps II, III, and IV.

Procedure

This final step of the methodology extends an aggregated STAPN model, made in a previous step, into a refined model of the same system. The purpose of this step is (a) to establish formal correspondences between the original and the disaggregated models, (b) to add detail to the refined model which was deliberately suppressed when the original model was built, and (c) to derive vertical relations between measures for the two models using the formal correspondences.

The ideal processes for refining a STAPN model for a system were established in subsection 2.7. Based on Step I and that discussion, the substeps required to extend a STAPN model are:

1. For each transition in the original model, determine whether the objects or messages crossing the boundary represented by that transition are to be decomposed into classes of objects.

2. For each transition in the original model, determine whether the boundary represented by that transition is to be decomposed into segments.

3. For each transition identified in substeps 1 or 2, insert a set of transitions in the refined model which capture the decomposition desired. For each transition not identified in substeps 1 or 2, insert a replica of that transition in the refined model.

4. For each place in the original model, determine whether the regions or facilities represented by that place are to be decomposed into subregions or subfacilities.

5. For each place in the original model, determine whether the timing model, decision rule, or processing step represented by that place is to be decomposed into segments.

6. For each place identified in substeps 4 or 5, insert an acyclic network of places and transitions in the refined model to capture the decomposition desired. For each place not identified in substeps 1 or 2, insert a replica of that place in the refined model.

7. For parallel paths, coordination mechanisms, or other detail not yet built into the refined model, add places, transitions, arcs, and initial tokens to enhance the disaggregated model.

8. Where objects must reside in regions or facilities for a period of time before moving on, define a timing model for the corresponding transition/place in the refined model.

9. Where objects may depart a region or facility along one of multiple paths, define a decision rule for the corresponding place in the refined model.

10. Where attributes may change as an object crosses a boundary, define an attribute map for the corresponding transition in the refined model.

11. For each measure defined in Step II for the original model, construct a vertical relation based on the correspondences set in substeps 3 and 6 here.

12. Go back to apply Step II to the new, refined model.

As in Step I, if the only objective is to identify behavioral measures, the substeps 8, 9, and 10 are optional.

Guidelines

The same general advisories apply to refinement of a model as to the construction of the top level model in Step I. In addition, it seems to be easier to delete existing model elements, which are inappropriate to the level of detail of a model, than to avoid inserting those elements in the first place. However, first attempts to refine a model tend to incorporate more detail than desired. Thus the natural construction of a refined model seems to include more details than necessary in the disaggregation and enhancement substeps, followed by an additional simplification substep where some of the new model elements are removed.

Also, after two or more iterations, the refined models tend to become quite unwieldy. A technique for managing the complexity of the model at each level is to partition the model into submodels. The natural location of partitions is at transitions, since these already model boundaries in the real system. The inputs to a transition which straddles a partition line appear in one submodel; the outputs appear in another. Care must be taken to ensure that each transition on a partition line is disaggregated consistently in each model in which it appears. Partitioning a model also permits some submodels to be refined while others remain unchanged.

In summary, the four guidelines for Step V of the methodology are:

1. Maintain clear connections between the each level of the STAPN model and any previously established views on the scope, purpose, or structure of the system.

2. Keep the each refined model as simple as possible, leaving details for later refinements.

3. Include more detail than necessary in the first attempt to build a refined model then eliminate elements which obscure the major issues addressed by the bulk of the additional model structure.

4. Partition models which exceed a few dozen places and transitions into submodels, with boundaries between submodels passing through transitions.

3.8.6 Conclusion

The five steps outlined in this subsection form the basis of a methodology for building hierarchical models and sets of measures. The design of the methodology endows the models and measures with certain desirable properties. While steps 3 and 5 generate some equations which relate measures to one another, these are insufficient to compute numerical values without further information about the system under study.

To conclude this subsection, some of the major advantages and disadvantages of the methodology presented here are reviewed.

Advantages

The advantages of the method stem from the structured decomposition which drives the model-building process, and from the mathematical structure of STAPNs. They include:

Reality: The connections established in Steps I, IV, and V between the STAPN elements and the objects, messages, facilities, regions and boundaries of the real system preserve a tight connection between the resulting measures and the activities of the real system.

Visibility: The hierarchical set of models, while initially motivated by a desire for a tool to help structure the generation of measures, is itself a useful product of the methodology. The models act as successively more detailed schematics of a system, and capture not only connectivity but also the timing and coordination mechanisms which operate within that set of connections. To an analyst who did not participate in the generation of the measures, the models make explicit a number of the assumptions that led to the selected sets of measures.

Automatic generation of measures: Because the models are built out of standard STAPN primitives, and the canonical measures can be readily deduced from the interconnection of these primitives, generation of the measures themselves becomes an easy task.

Automatic minimality: Along with the canonical measures come canonical horizontal relations, which establish mathematical redundancies among the measures. Eliminating a subset of the measures so that no relation holds among the remaining, primary measures assures independence.

Automatic internal completeness: The formal nature of disaggregation at transitions and places establishes the one-to-one correspondences required for vertical relations to hold. The fact that a vertical relation exists for each higher level measure ensures that the values of all higher level measures are completely detemined by

values for lower level measures. Thus nothing is missing from the lower levels, at least with respect to the coverage of the measures at the higher levels.

Precision: The rigorous definitions of measures for STAPN models guarantee that the theoretical definition of every measure constructed using this methodology is precise. The general analogies provided by Assertions 1 - 4 in Section 3, and the glossary provided in step 4 of the methodology enable the mathematical definitions to be translated into precise and unambiguous physical definitions.

Disadvantages

However, a number of disadvantages prevent the methodology from achieving all of the goals one might expect of it. Four major disadvantages are:

Judgment: The selection of the amount of detail to be preserved at a model level, and of the way to disaggregate higher level elements into lower level structures, is completely determined by the judgment of the analyst executing the methodology. Since different analysts have different opinions, the models and measures produced by two independent studies of a single system are likely to be different.

Incompleteness: Many details of a system are deliberately suppressed at the higher model levels for the sake of intelligibility. These details may be added as the depth of the model hierarchy is increased, but practical constraints on time and effort may limit this depth. Thus the resulting sets of measures may be knowingly incomplete. In addition, there seems to be no way to exclude (external) incompleteness caused by simple oversights.

Structural and socioeconomic measures: The focus of the development here has been on behavioral measures. Structural measures may fit into the STAPN framework if tokens are taken to represent structural states, but deserves further thought. Socioeconomic measures are well outside the capabilities of the STAPN approach.

Complexity: Above all, C^3 systems are complex. Models and measures developed to describe a system will also be complex. Complexity engenders frustration, impatience, and skepticism. Nonetheless, complexity is a price that must be paid by any methodology which aspires to completeness and realism.

3.9 IAT QUESTIONS THAT MAY BE ADDRESSED

In Section 1 were listed questions about C^3 systems that should be addressed by the Integrated Analysis Techniques summarized in Fig. 3-10. It is

now possible to determine which of these questions can currently be addressed, and which must await further technique development. Each question is repeated below for convenience, followed by a brief discussion of its addressability.

1. Given a static structural description of a C^3 system, how can one predict the system's performance?

   Answer: Referring back to Fig. 3-10, evaluation of predicted C^3 system performance measures requires that a STAPN process model of the system be produced and that all of the requisite data identified in the left-hand column of Fig. 3-10 below the horizontal dashed line be obtained and reflected in various model parameters, at a selected level of system decomposition.

2. What can a static structural description tell one about:

   - the strengths and weaknesses of the way functions are performed (i.e., by the mechanisms or resources which carry out the functions)?

   - the strengths and weaknesses of the way functions are combined (i.e., carried out by the same resource)?

   - the dependency of functions (i.e., upon other functions, resources, etc.)?

   - the strengths and weaknesses of data flows and controls (i.e., functional connectivity)?

   - the criticality of functions, data flows, mechanisms, and controls?

   Answer: Given a STAPN process model, a complete set of performance measures can be defined. However, they cannot be evaluated unless the requirements noted above are met. Determining the relative strengths and weaknesses of the way in which various functions are performed and of data flows and controls, as well as determining the relative criticality of these items, requires such evaluation. On the other hand, the dependency of functions upon other functions, resources, etc., can be determined directly from the appropriate matrix in Table 2-1. Even second, third, and n-th order dependencies within a given level of decomposition can be calculated by means of simple matrix multiplication. For example:

Let $[P^{L}_{i} \times P^{L}_{j}]$ be the functional connectivity matrix defining which processes P_i feed which other processes P_j at the L-th decomposition level. This is a sparse matrix of zeros and ones which directly represents the first-order dependencies among the processes. Then $[P^{L}_{i} \times P^{L}_{j}]^2$, $[P^{L}_{i} \times P^{L}_{j}]^3$, and $[P^{L}_{i} \times P^{L}_{j}]^n$ represent the second, third, and n-th order dependencies, respectively. Note that a STAPN process model is not necessary for this purpose; the required matrices can be developed directly from any appropriate functional or resource connectivity diagram.

3. How can one use a static structural description, along with any other transformations, augmentations, or other data, to answer the questions in item 2 above? What measures can be used?

   Answer: This question has already been addressed by the answers to questons 1 and 2 above.

4. What can the static structural description tell us about the dynamic performance of the system? How does it address or support issues of:

   - timeliness
   - probability of error
   - survivability?

   Answer: Dynamic performance can only be determined by exercising a STAPN process model with all of the requisite data, as in the answer to question 1. The "PROD" statistics will directly provide probabilities, rates, occupancies, and delay data. Calculation of error probabilities and their consequences will require that the STAPN model explicitly include branches for representing critical error possibilities and their propagation paths.

   Static functional survivability measures can be defined and evaluated for a given C^3 structure and attack scenario, using the methods of Wohl et al. (Wohl et al., 1981). However, they cannot be evaluated for a dynamic situation in which the system structure continually changes as a result of jamming and/or destruction. New measures and methods must be developed to handle such situations.

5. What do classical systems engineering theory, organization theory, or network theory offer in the way of properties or measures to address the foregoing issues?

   Answer: A STAPN process model represents the most recent advance in systems engineering theory. Simpler STAPN models with single-target scenarios can be evaluated using analytical methods from PERT/CPM, and with multiple-target scenarios they can be evaluated using methods from queuing network theory. More complex STAPN models (e.g., at greater levels of decomposition detail) must employ computer simulation for their evaluation. Organization theory will be useful in the future in developing methods for analyzing dynamically adaptive (i.e., changing) C^3 structures and in calculating their survivability measures (see 4 above).

6. How can the answers to the above questions be used to improve system performance?

   Answer: At present this can only be done by comparative evaluation of alternative structures, alternative assignments of resources to processes, etc. At some future date it should become feasible to apply dynamic optimization techniques to determine how a system should be structured and what are the optimal allocations of resources (including humans) to processes. However, these techniques are not yet sufficiently mature to be able to address hierarchical C^3 system structures.

SECTION 4

ANALYTIC METHODS FOR EVALUATING C^3 SYSTEM PERFORMANCE

4.1 TOOL SELECTION AND SPECIFICATION

A top-down technical approach was done, first examining properties of operations research and systems engineering tools, and then evaluating them for relevance to the kinds of C^3 evaluation problems of which the NORAD MWC operational environment is typical.

The most important results are as follows:

1. For limited application, e.g., to help gain insight into more complex systems, critical path analysis based on standard PERT/CPM methods can be useful. Its primary limitation is that it will handle only the passage of a single threat through the system; but even this can demonstrate the presence of major bottlenecks in a manned C^3 system. Because of their general availability (e.g., Boehm, 1981), analytical details are not repeated in this report.

2. Analytic approaches based on queuing theory can indeed be used to exercise performance models of manned C^3 operational systems under more realistic (i.e., multiple target) conditions, but only if sufficient care is taken to modify classical queuing theory for describing and predicting human performance. Again, standard queuing theory techniques are generally available and will not be repeated here (Kleinrock, 1976). Appropriate modifications for application to human performance are described in subsection 4.3.2.

3. Petri nets, and in particular, STAPNs appear to be especially useful as a means of:

   a. representing PROCESSES and RESOURCES within the IAT framework;

   b. constructing performance models; and

   c. defining measures of performance (MOPs)

that will permit the models to be exercised. These aspects of STAPNs have already been described (Section 3 and Appendices A and B). Functional and data requirements for using Petri nets are elaborated in subsection 4.3. It should be noted that any queuing model can be represented by a Petri net.

4.2 PERT/CPM TECHNIQUES

The identification bottlenecks in any flow network can most easily be done using PERT/CPM methods if (1) the network's nodes and arcs are completely defined; (2) the processing delays at each node are known, either deterministically or statistically; (3) we are only interested in the single-thread case, e.g., a single intruder in an air defense system. PERT/CPM techniques allow simple and direct computation of the total expected delay along any set of connected paths and, as a fallout result, the longest path delay or "critical path" (hence the term Critical Path Method or CPM). The computational methods are well known (Boehm, 1981) and have been embodied in many computer programs (including several for PC's). As a consequence, they will not be repeated here.

4.3 QUEUING THEORY APPROACHES

Studies conducted during FY82-84 identified queuing theory as an appropriate method for generating quantitative estimates of human/system performance in C^3 systems:

1. It can be used to produce measures of throughput and delay directly related to the timeliness measures natural to C^3 systems.

2. Measures of resource utilization are provided. (These furnish the means to explore alternative resource allocations and task structures.)

3. The analyses can be done at several levels of detail (ranging from gross approximations based on simplifying assumptions and steady-state analyses to detailed transient analyses obtained with numerical methods).

4. Issues of accuracy and error can be addressed through the parameters of simple queuing models or their extension, a network of queuing models.

In this subsection we present a brief review of classical queuing theory based largely on Kleinrock (1976) and summarize the assumptions that need to be made for analyzing and predicting human performance. Modifications to simple queuing analysis methods are required because these traditional approaches

are not designed to take account of human behavior at the level of complexity exhibited in C^3 systems. Changes to standard methods are motivated for addressing factors such as accuracy and error, associated with human operators who carry out specific tasks at workstations.

4.3.1 Queues and Their Relevance to Modeling C^3 Systems

Queuing theory involves the mathematical characterizations and analysis of "queues" (waiting lines). Queues form whenever demand for a service exceeds capacity to provide that service. In real-world systems, units of demand are items-to-be-serviced, and may take the form of messages, information, raw materials, or tasks that need to be processed. Entities providing service or processing may be human operators, computer programs, or entire C^3 systems.

Decisions must be made regarding the amount of capacity, or resources that should be allocated to maintain systems that will function in a cost-effective manner. To allocate resources appropriately and to reduce costs, decision-makers would like to be able to predict when units (of demand) will arrive to seek service and how much time will be needed to provide the required service. This information becomes important for achieving a balance between the cost of providing a service and the cost associated with waiting for that service. Providing too much service (i.e., more than required to handle demand) creates unnecessary expense; but not providing enough service causes queues to build up beyond processing capacity.

Queuing theory analysis does not itself tell decision-makers how to balance supply of service against costs of waiting --but it does yield the data needed to make decisions by predicting characteristics of the waiting line (e.g., mean waiting time, length of queue).

Queuing Models: Basic Structure

Figure 4-1 represents a simple queue. Items to be processed ("customers") originate from an "input source" (or "input population"), and enter the "queuing system" when they join a queue. At specific times, a customer is selected for service according to a rule called the "service" or "queue discipline;" this discipline refers to the order in which customers are selected for service. The "service mechanism" performs the required service for the customers, after which they leave the queuing system.

Elements of the Queuing Process and Standard Assumptions

1. INPUT SOURCE -- Population size (total number of customers that may require service) is assumed to be infinite

   -- Customers are identical

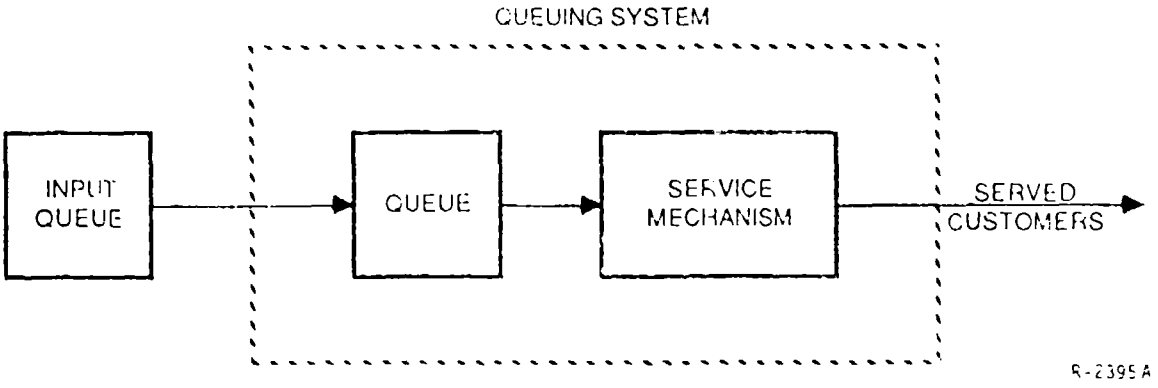


Figure 4-1. The Basic Queuing Process

-- The number of customers generated until a specific point in time has a Poisson distribution

(for the case where arrivals to the queuing system occur randomly but at a certain average rate)

-- The interarrival time (time between consecutive arrivals) follows an exponential distribution

-- Unless otherwise stated, there is no balking or reneging (customers refuse to enter the queue or may leave if the queue grows too long)

2. SERVICE DISCIPLINE -- First-in/first-out ("FIFO") unless specified to the contrary

3. SERVICE MECHANISM -- Consists of service facilities, which are comprised of service channels ("servers"). Most elementary models assume one service facility with either one or a finite number of servers.

-- "Service time" ("holding" or "completion time") is the time lapsed between start and completion of service; the probability distribution of service time is assumed to be exponential and the same for all servers. Constant average service time is frequently assumed. Service time does not depend on the attributes of a particular customer.

4.3.2 Using Queuing Theory Approaches to Model Human Performance

Notational Conventions for Modeling Human Performance

For purposes of using queuing theory to model human performance in manned C^3 systems, the notations below describe a classical "M/M/1" queue, for a single human (server), performing tasks (handling customers) of a single type. In an M/M/1 queue:

- M/M/1 (the first "M") denotes the arrival process type; viz., Poisson, having a constant average task arrival rate (λ tasks/sec.) for all arrivals.

- M/M/1 (the second "M") describes the service process type; viz., exponential, with a constant average task completion rate per busy server (μ tasks/sec.).

Note: the exponential interarrival and service time distributions mean that the system is "memory-less" -- i.e., at any point in time, the total length of the time in the system is independent of the observed line length.

- M/M/1 (the last integer) identifies the number of servers.

The queuing process is assumed to be steady-state; i.e., it is defined with respect to the average time between arrivals, average service time, and the queue discipline. From these parameters, statistics can be derived to describe time in the queue, time in the system (queue plus processing), number of customers in the queue, and idle time of the system or service facility. Table 4-1, which follows, presents the symbology and notational conventions that are consistent with these assumptions. Table 4-2 includes formulas to describe the queuing process for elementary queuing models.

Limitations of the Simple Queuing Model

In characterizing the "M/M/1" model, we have assumed that --

1. (Human) error rates are not taken into account.

2. Service rates are independent of arrival rates.

3. Waiting times are not constrained (i.e., tasks will wait forever if necessary to be completed).

Each of these three assumptions needs to be changed for using simple queuing models to characterize and predict human/system performance. The following sections discuss changes appropriate for modeling human operator performance.

TABLE 4-1. STANDARD TERMINOLOGY FOR STEADY-STATE SIMPLE QUEUING MODELS (Kleinrock, 1976)

n	Number of customers in the queuing system
P_n	Probability that exactly n customers are in the queuing system
L	Expected number of customers in the queuing system
L_q	Expected number of customers in the queue
W	Expected waiting time in the system (includes service time)
W_q	Expected waiting time in the queue (excludes service time)
λ	Mean arrival rate (expected number of arrivals per unit time) of new customers
$1/\lambda$	Expected interarrival time
μ	Mean service rate (expected number of customers completing service per unit time)
$1/\mu$	Expected service time
ρ	Utilization factor (expected fraction of the time that servers are busy), defined as λ/μ

TABLE 4-2. FORMULAS FOR DESCRIBING THE QUEUING PROCESS

For a steady-state queuing process:

$$L = \lambda$$

and

$$L_q = \lambda W_q .$$

For a classical M/M/1 queue:

$$L = \frac{\lambda}{\mu - \lambda} = \frac{\rho}{1 - \rho}$$

therefore,

$$W = \frac{L}{\lambda} = \frac{1}{\mu - \lambda} .$$

[May also be designated as:

$$E(t_b) = \frac{1}{\mu - \lambda}, \text{ the expected length of the busy period)}$$

where

$$E(t_i) = \frac{1}{\lambda}, \text{ the expected length of time the server is idle; i.e., the expected interarrival time.]}$$

However,

$$W = W_q + \frac{1}{\mu}$$

therefore,

$$W_q = \frac{1}{\mu} \left(\frac{\rho}{1 - \rho} \right) = \frac{\rho}{\mu - \lambda}$$

and $$L_q = \frac{\rho^2}{1 - \rho} .$$

It can also be shown that:

$P_o = 1-\rho$ (the probability that the queue is empty) .

Error Rate Models

In the simplest case, there is an error probability $P_e(\rho)$ as a function of ρ, defined as the probability that a task is incorrectly completed. In the simple case, incorrectly completed tasks are not redone. In a more complex case, the incorrectly completed tasks are redone. Consequently, for the more complex case, the rate λ of tasks is defined as

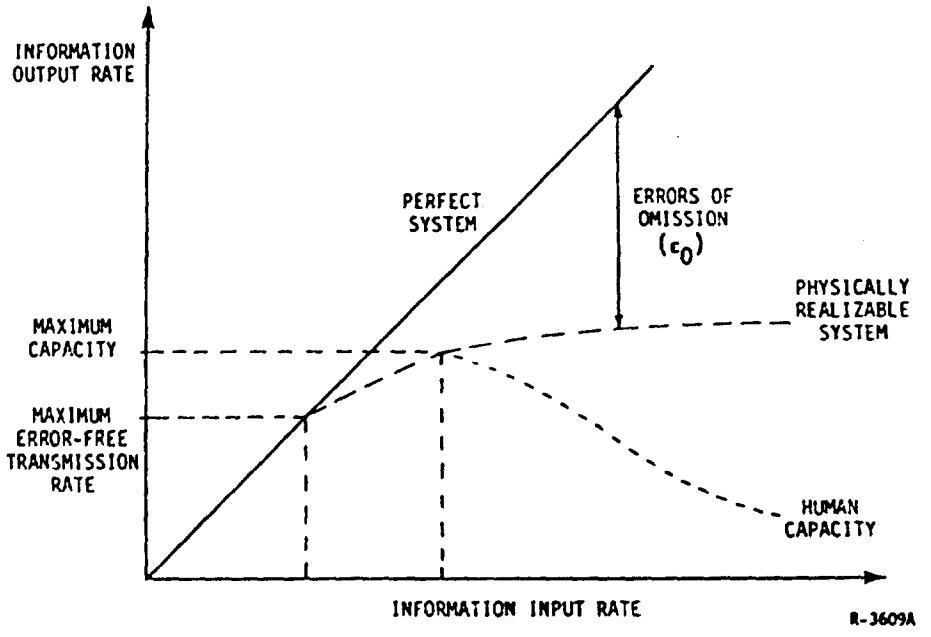
$$\tilde{\lambda} = \lambda + \lambda P_e \left(\frac{\lambda}{\mu}\right)$$

where λ is the externally-imposed task arrival rate and the second term is due to the feedback (i.e., redoing) of incorrectly completed tasks. Note that such a "human" queue, in contrast to the classic M/M/1 queue, will saturate before the utilization factor $\rho = \frac{\lambda}{\mu}$ is equal to 1. In fact, it can be shown that the human saturates for $\rho = 1 - P_e(1) < 1$.

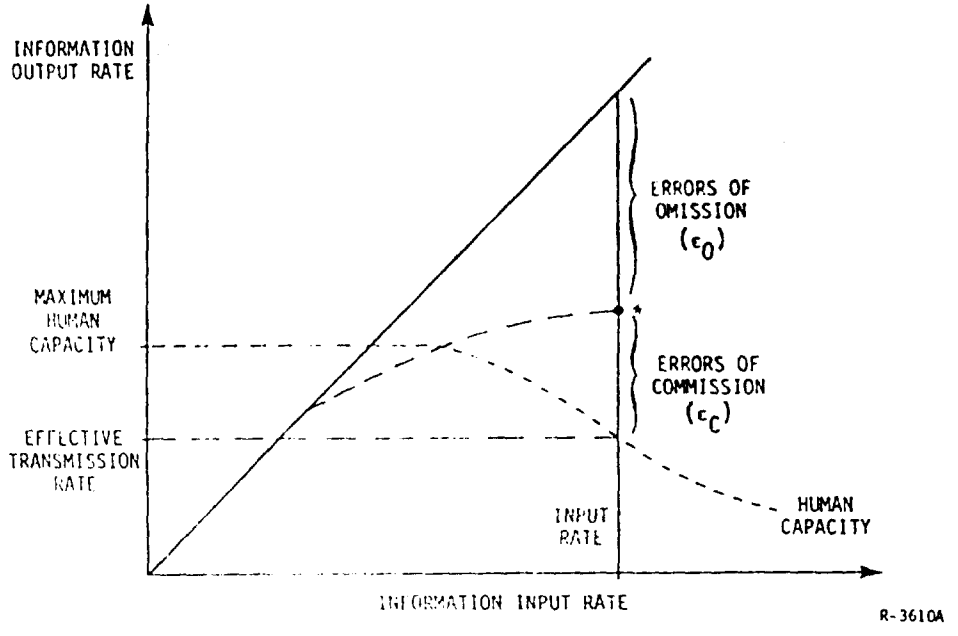
Of course, it is necessary to have an empirical database to determine the function $P_e(\rho)$, or even just a constant value of P_e. Moreover, the task for which a queuing model with errors is used must be such that task completion can be characterized simply as correct or incorrect. Other formulations are possible, such as distinguishing between errors of commission and errors of omission (Schank and Abelson, 1977).

Errors of Commission. Errors of omission occur as a result of the relationship between capacity, mean arrival rate (λ), and expected waiting time (W), as in Fig. 4-2a. For human operators, we suspect that the mechanism of error generation also involves an attempt to adjust or "stretch" response capacity C such that waiting time W of a task does not exceed a criterion level. Then, as apparent capacity C increases (e.g., as in the uppermost horizontal dashed line in Fig. 4-2b), the rate of errors of commission ε_C (e.g., character substitution, number inversion, types, etc.) should increase proportionately, in accordance with Shannon's law for noisy communication channels. Specifically, for $\mu < C$, error rate ε_C can be made arbitrarily small by proper task design (i.e., "encoding"). But for a given task design, any attempt on the part of the human to increase his apparent capacity C will result in an increase in error rate ε_C.

Errors of Omission. By adding a threshold parameter to the classical M/M/1 model, we can predict errors of omission. Assume that if W for a given task exceeds a threshold parameter τ, then the task is omitted by the serving resource and we say that a "reneg" has occurred; i.e., the "customer" or task cannot "wait" any longer than a threshold time $W = \tau$ and "leaves" the queue. This reneging rate is equivalent to the error rate for errors of omission, ε_O, such as missed keystroke, missed symbol, missed message, etc.



(a) Differences Between Perfect, Physical, and Human Information Handling (after Miller, 1978)



* ATTEMPTS TO PROCESS FASTER RESULT IN MOVING THIS POINT UPWARDS; i.e., FEWER ERRORS OF OMISSION BUT MORE ERRORS OF COMMISSION. TOTAL ERROR RATE REMAINS CONSTANT.

(b) Effect of Human Attempts to Process Faster

Figure 4-2. Human Error and Workload - An Information-Theoretic Paradigm

Let τ be scenario-determined "time available" for a task from queue entry to completion of service; and let $1/\mu$ be the "time required" for service. Then τ must be greater than the service time $1/\mu$ plus the waiting time W in order to minimize reneging. That is, in the steady state,

$$\tau > (W_q + 1/\mu) \quad .$$

If τ is less than this value, reneging or errors of omission will occur at an increasingly high frequency.

The "slack time" S for a task is the time available minus the time required for the task. That is,

$$S = \tau - 1/\mu = 1/\lambda - 1/\mu \quad .$$

For a lower bound for this expression, we substitute for τ from the foregoing expression to obtain:

$$S > \frac{1}{\mu} \left(\frac{\rho}{1 - \rho}\right) \quad .$$

Workload-Dependent Service Rate

Let us assume that the service rate depends on the utilization factor,

$$\mu = \mu(\rho) \quad .$$

Then the effective service rate $\tilde{\mu}$ for a given queue must be derived from the expression

$$\tilde{\mu} = \mu\left(\frac{\lambda}{\mu}\right) \quad .$$

Note that an alternative assumption would be that the human's processing rate is a function of the actual backlog of tasks to be performed, rather than average operator utilization.

Reneging

An additional consideration is the potential reneging of tasks, i.e., tasks that are not completed by a certain deadline cannot be completed at all.

An example is a task (related to ballistic missile intercept) that cannot be performed if the incoming weapon impacts its target. As discussed above, non-completion of tasks can be viewed as errors of omission, and reneging/error rates predicted.

4.3.3 Functional and Data Requirements for Queuing Theory Approaches

Lessons Learned from the SIMCOPE Modeling Experience: Defining Queues

The analysis that was done on the SIMCOPE simulated missile warning facility at AAMRL (see Vol. II) provided experience for dealing with the issues, problems, and expected benefits of using queuing theory approaches to describe and predict human performance. The primary technical issues that had to be addressed for SIMCOPE (and for other similar operational environments) are the following:

1. What is the appropriate level of abstraction for describing human/system processes?

2. If a queuing (network) representation is to be used, what is it that will be queued and serviced? (viz., data, information, physical objects, or tasks?)

In dealing with (1) above, it has been assumed that IAT structural modeling must be taken down to a level of detail sufficient for identifying elemental tasks. At this level, measures of performance (MOPs) can be associated with specific processes that particular individuals carry out and for which they are responsible. Although measures may well be associated with higher levels of analysis (in aggregated form), the raw data about human performance should be collected at the tasking level.

Question (2) then must be considered. One option is to let tasks be queued to a human operator (or other resource) and serviced. This approach was not taken in modeling SIMCOPE for the reasons listed below:

1. Strict definitions of "tasks" are not easy to formulate.*

2. Completing a task might require several decisions and/or actions on the part of the operator, and these might be difficult or impossible to observe.

3. The procedural controls placed on an operator make task queues troublesome to describe.

*However, it is possible to provide a formal description of the level at which tasks can be meaningfully identified. Set theory notation can be used to specify appropriate level(s) of detail, as part of the IAT recursive decomposition technique described in Section 2.

Specifically, if tasks in the SIMCOPE example were comprised of message-handling processes (acknowledging, assigning, and filling out reports), it would have been very difficult to place these tasks in a single queue -- or even parallel queues based on separate task streams. The source of the problem lies in the interrelations among tasks. Even for the relatively simple environment in SIMCOPE, the number of interrelated tasks was large. For real-world systems, the situation is aggravated.

Another option, and the one that was taken in modeling SIMCOPE, requires that queues be defined in terms of physical entities (IAT RESOURCES). Then the processes necessary to service the items in the queue can be considered in a more straightforward manner. The chief merit of this approach is that it places the source of the demand for service with items which are easily identified and quantified -- rather than with tasks which may be somewhat more open to interpretation.

One consequence of defining queues in terms of RESOURCES is that a data flow analysis identifies such queues directly. Moreover, any situation where items are filed (or stored) in anticipation of further processing suggests a queue. Hence, there is no need to define tasks exhaustively before identifying queues. Another advantage to this approach is that measurement issues and data collection become simplified: queues are based on observable physical items and events, as opposed to perceptual or cognitive ones. This last point becomes critical when descriptions of the system (such as the classified/unclassified documents on NORAD MWC and CP) do not contain service time data for macroscopically defined tasks. This was the situation in SIMCOPE and would appear to be the case for the NORAD MWC Validation Effort as well.

It is essential therefore that data necessary to support or validate the queuing theory approach be directly observable or capable of being derived easily.

Defining Queue Discipline

An independent reason for using resources to comprise queues comes from examining problems associated with defining queue discipline. In the case of a human operator (or any resource), goals and procedures will define what task should be performed in a given circumstance. A task analysis should then identify the data used by the operator, as well as those processes the operator carries out to complete the task. This type of analysis will define the queue discipline albeit indirectly, but the description will be a natural one from the prospective of depicting human performance.*

*This approach was illustrated in the SIMCOPE example via the overall data flow diagram and the estimation of effective processing times. The diagram identified services or tasks performed by the operator while the service time estimates were dependent upon the assumptions about the order of processing and interruptions.

4.3.4 Recommendations for Using Queuing Theory to Evaluate Human/System Performance

1. View the human operator as a server for one or more queues.

2. Define these queues by tangible (observable and measurable) items in the system. (IAT RESOURCES)

3. Use a data flow description (e.g., with DFDs) to facilitate identifying queues and major processes. (This will allow the queuing network to be described with a minimum of abstraction.)

4. Make explicit the control structure of the system. (This is especially important for cases in which one operator or resource services more than one process. A description of the system control structure would establish the order in which processes are to be performed.)

5. Use Petri nets to specify control structures.

6. Carry out further analysis (process modeling) to estimate service rates:

   a. Identify processes shown on the lowest-levels of DFDs. These processes are the appropriate ones for completing a given task, according to the conventions of data flow methodology.

   b. Determine completion times for each process identified in 6a).

   c. Estimate conditional task completion times ("conditional" in the sense that full attention is assumed to be devoted to the task).

Levels of Detail

The SIMCOPE example and other experience suggests that while some of the impact of system connectivity and structure on human function allocation could be analyzed at higher levels of aggregation, the level of process description defined in Gruesbeck et al. (1984) was not sufficiently detailed to derive the quantitative data necessary for performance analysis. Further decomposition must be performed and data must be extracted by considering procedures and system specifics (e.g., cued versus non-cued display, etc.). Such data are obtained only after fairly detailed task analysis is completed. The decomposition should push down to the point of revealing generic processes such as button-pushes and extraction of single items of data. This will insure that analysts can use information from the human factors literature to fill

data voids in system-specific documentation. By working back up through the process description, the necessary service time data can then be estimated.

While gross trial allocations of functions can be performed at higher levels of aggregation, only if given sufficient detail is it possible to link human factors design issues and data into a queuing network representation for predicting performance. Without going to this point, it is not clear that such linkage can be achieved except in a purely empirical way (i.e., by estimating parameter values from observations of the system in operation). Also, it is only at this level that operator interface design and redesign issues can be addressed.

Addressing Issues of Accuracy and Error

As illustrated in the SIMCOPE analysis in subsection 4.3.3, error rates can be established by the task or service description. To use this approach, the process description must be sufficiently detailed so as to describe error-checking, editing, and various exit conditions* in a probabilistic manner. Service times will then directly reflect error by means of checking and correction loops: these will have the effect of increasing average service times. Problem simplification strategies and load shedding could also be built into the detailed process description, if desired, by making the processing activity state-determined.

4.4 STAPN MODELING

In Section 3 and Appendix A a detailed basis for STAPN modeling of C^3 systems, including humans, are provided. However, there are certain minimum data requirements for using Petri nets which bear repeating at this point.

4.4.1 Summary of Basic Data Requirements

To describe a model of PROCESSES, from which measures of performance (MOPs) and measures of effectiveness (MOEs) may be computed, at least the following is necessary:

1. A list of all of the places.
2. A list of all of the branches.
3. A list of all of the transitions.

*e.g., -- Task completed error-free
-- Task completed with one error
-- Task completed with multiple errors
etc.

4. For each place:

   a. A description of the process by which unavailable tokens are made available.

   b. The transitions which insert tokens into the place.

   c. The branch which describes where tokens go after they leave the place.

   d. The number of tokens initially in each place.

5. For each branch:

   a. A description of the decision rule which selects the path each token will follow.

   b. The place from which tokens exit through the branch.

   c. The transitions to which tokens may flow (i.e., the alternative paths leaving the branch).

6. For each transition:

   a. The places which must contain available tokens for the transition to fire.

   b. The places which receive unavailable tokens when the transition fires.

These requirements can be grouped into three classes:

1. Decision rules (5a)

2. Timing data (4a)

3. Network topology (all others).

These form three basic "dimensions" of the physical process model.

4.4.2 Metrics for Insuring Model Quality

One of the major outputs of an IAT analysis is a physical process model from which performance measures will be calculated. At least four desiderata apply to the output of any such methodology:

1. Consistency: Is the model (formally) well-structured? Are all connections legal?

2. Completeness: Is the model finished? Are there any loose ends that need to be filled in?

3. Correctness: Is the model an accurate representation of reality?

4. Parsimony: Is the model represented at an appropriate level of detail?

Consider the implications of each in a Petri net framework:

1. Consistency

A Petri net model has a strong syntax: places lead to branches, which lead to transitions, which lead to places. Any model which violates this syntax, e.g., by connecting a transition to a transition, is inconsistent (Figure 4-3). While such inconsistencies can be detected by a "model syntax checker," like the syntax checker in a compiler, it would be preferable to have a methodology which ensures that the resulting model is consistent.

2. Completeness

A physical process model is complete if it contains enough information to allow it to be simulated. For Petri nets, this means:

a. every branch is preceded by a place

b. every transition is preceded by at least one branch

c. every place has an attached process (timing) model

d. every multiple output branch has an attached decision rule

e. every place has an initial number of tokens specified (default may be zero).

Any consistent network description satisfying a) and b), with ancillary information (c), (d), and (e), can form the basis of a simulation.

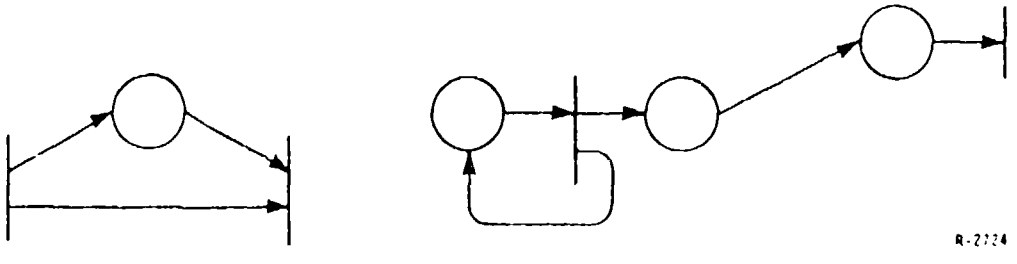


Figure 4-3. Inconsistent Petri Net Models

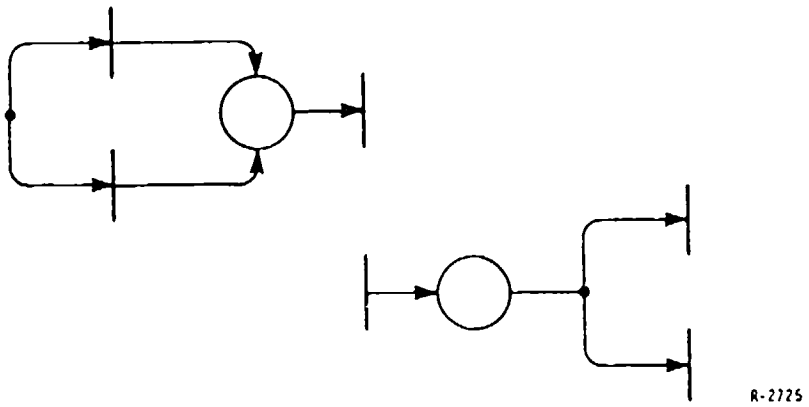


Figure 4-4. Incomplete Petri Net Models

3. Correctness

Unlike consistency and completeness, correctness cannot be assessed just by examination of the model. The usual issues of validation, etc. cannot be avoided. However, there are symptoms of incorrectness which should be double-checked when found:

a. A place with no inputs and no initial tokens is entirely superfluous.

b. A place with no inputs and some initial tokens is of transient importance only.

c. A transition which can never be enabled is superfluous.

Related to these symptoms, "liveness" (related to c)) and "boundedness" (no place where an infinite number of tokens can accumulate) have been extensively studied in the literature. Dead or unbounded Petri nets may be incorrect models, or may be correct models of improperly designed systems.

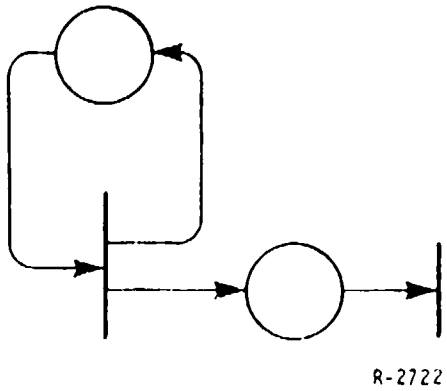


Figure 4-5. Incorrect Model: "Dead"

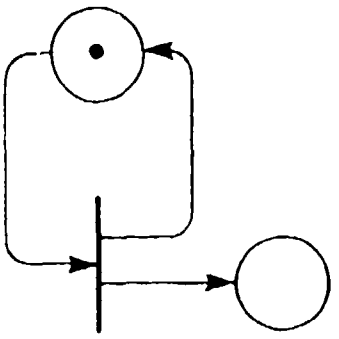


Figure 4-6. Incorrect Model: "Unbounded"

4. Parsimony

This is an aesthetic quality. There is a fundamental tradeoff between the complexity of processes and decision rules and the complexity of the network.

4.4.3 Extensions for Enhancing Model Clarity and Completeness

1. Annotations - text commentary indicating the relationship of a model element to the actual system.

2. Token Typing - labels on tokens to describe the function/object/data they represent.

3. Multilevel Representations - aggregation of the process, protocol, and timing information into higher-level constructs (which perhaps have non-local dependencies).

4. Other Dimensions - physical equipment models (from which timing information can be derived), organization/goal models (from which protocols can be derived), etc.

4.4.4 Summary

In summary, Petri net process modeling can be completely described by the items in subsection 4.4.1 plus

1. An annotation describing the facility or resource that the place represents.

2. An annotation describing the purpose or goal of the branch.

3. An annotation describing the event modeled by the transition, including uncertainty, timing, and dependency relationships.

4. For each token type,

   a. An annotation describing what the token represents.

   b. The transition(s) at which the type is created.

   c. The transition(s) at which the type is destroyed.

5. As appropriate, aggregate models and token typing hierarchies.

SECTION 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 CONCLUSIONS

While numerous system representation methods such as $IDEF_0$, Data Flow Diagrams, and the SAINT and other simulation languages have long been available, none of these techniques, either alone or in combination, could meet the IAT requirements noted in Section 2. The main obstacle was the lack of a single, underlying, integrating analytical framework (i.e., a theory of C^3). With the development of the Stochastic, Timed, Attributed Petri Net STAPN) representation technique for manned C^3 systems, this obstacle has now been largely overcome.

The results of the work to date have served to demonstrate the feasibility of IAT. Not only has the required analytical framework for IAT been developed, but the following additional results have also been achieved:

- a mathematically rigorous symbolic language (STAPNs) for describing (i.e., modeling) and evaluating (i.e., assessing the system performance and/or military effectiveness of) manned C^3 systems and their associated weapon systems at any level of description or decomposition.

- a convenient means for managing system complexity by aggregating and modularizing system details in the new Petri net "Box Node" aggregation prmitive, without masking the impact of these details on overall system performance and effectiveness.

- a set of nested and self-consistent system measures derived from the symbolic language.

- a flexible data management system concept based on the artificial intelligence concept of frames and slots.

- computer-based instantiations of all of the above.

In Sections 3 and 4 it has been shown that classical analysis methods such as PERT/CPM and queuing theory, as well as computer simulation methods, can be made an integral part of the STAPN methodology for IAT.

5.2 RECOMMENDATIONS

Based on the lessons learned from the three trial applications discussed in Vol. II, it is felt that an automated aid to IAT must be developed with the following features:

1. A "Friendly Front End (FFE)" to help the analyst describe and decompose a C^3 system using a "standard" graphic input language of his or her choice. Specifically:

   - An analyst must be able to "enter" IAT using familiar or easily learned, graphics-based techniques such as Data Flow or $IDEF_0$ Diagrams, and the FFE must interactively assist the analyst in generating, decomposing, storing and accessing these diagrams.

   - The FFE must incorporate a "Data Stripper" or means for automatically stripping off from the stored Data Flow and $IDEF_0$ Diagrams the connectivity, information flow, dependency, control, resource assignment, and other data required to convert it into a STAPN model.

   - The FFE must also incorporate a "Data Checker" to help the analyst build the required database. The mathematical rigor of the STAPN modeling method automatically provides a "model" of the relationships among the various data elements, thus greatly reducing the difficulty of Data Checker development. The Data Checker must work interactively with the analyst to flag missing and/or inconsistent data and connectivities, and must also help generate a list of C^3 resources and their assignments to C^3 processes.

2. A database system organized around the Frame/Slot concept of artificial intelligence to provide the flexibility needed to make possible the features described herein. Data organization is crucial to the success of an automated aid to IAT. The data must be organized dimensionally and hierarchically to reflect the decomposition levels within and among each of the four descriptive dimensions (Process, Resource, Organization, and Goal). For example, slots in a given process frame must be capable of being used as "pointers" to associated processes, to assigned resources, to organizational assignments, to assigned performance goals, and to associated sets of measures (e.g., "PROD" statistics) contained in other frames, while maintaining consistency of description among the dimensions at a given decomposition level. To reduce the effort involved, the system should also contain a "default" database of key C^3 system and human operator/decisionmaker parameters. This will permit sensitivity analyses to be performed to determine which data item values require further refinement (e.g., by means of man-in-the-loop experiments).

3. An "Explainer" for explaining to the analyst why certain data inputs were required, why the structure of a STAPN model was incomplete or inconsistent, and why selected MOPs and MOEs were obtained when the model was exercised. Without such a feature, we have found that analysts will be loathe to use the automated aids described above, and their "customers" will question the credibility of their results.

Fortunately, except for the Data Stripper, many of these features have already been or are in process of being developed for other projects, and need only to be combined and integrated to form an automated analyst's aid to applying IAT. It is recommended that this be done.

APPENDIX A

MATHEMATICAL FORMALISM FOR STAPNs

A.1 INTRODUCTION

In this appendix we present details of Stochastic, Timed, Attributed Petri Nets (STAPNs). The canonical Petri net variables and their relationships are identified along with methods for both their decomposition and aggregation. Techniques for evaluation of measures are briefly described (detailed examples will be found in Volume II), and Appendix A ends with a description of some of the systems analysis issues that have not yet been addressed.

A.2 CANONICAL MEASURES: PROD STATISTICS

In subsection 3.4 we established tight connections between the reality of a C^3 system and the structure of an abstract STAPN representation of that system. Given these high fidelity relations, conclusions drawn from the STAPNs carry over directly to analogous statements about reality. In particular, this section explains how the STAPN primitives give us a way to identify canonical variables which describe their behavior, and henceforth we take for granted the fact that these variables apply equally well to a real system. Appendix C contains procedures and guidelines for generating measures from STAPNs.

There are four types of canonical measures, each associated with a different element of a STAPN: arcs, transitions, places, and tokens. There is no distinction between capability measures, mission measures, or effectiveness measures: each of the four canonical types can play any of the three functional roles. In order to prod one's memory about the forms of the canonical measures, we consider them in mnemonic order (probabilities, rates, occupancies, and delays), and do not distinguish between roles.

Before introducing the canonical variables themselves, two technical aspects of their definitions must be settled. First, some basic stochastic processes associated with STAPNs are introduced, processes that will form the basis of explicit evaluation mechanisms alluded to in Assertion 4 on page 60. Second, each basic measure may be defined in four different ways, depending on the evaluation mechanism used.

A.2.1 Basic Processes

Begin by considering any arc A_i in a STAPN. Suppose the STAPN's operation is started several times, each execution called a replication of the process. For each replication, observe the tokens that flow along A_i. These tokens flow instantaneously and in sequence (ties may be broken arbitrarily to establish a total order on the tokens that follow A_i). Let $t_i(n, k)$ be the time (according to some external, global clock) that the n-th token traverses A_i during the k-th replication of the STAPN operation. The $t_i(n, k)$ form a set of random variables which characterize much of the STAPN's behavior.

To derive measures from the $t_i(n, k)$, for each arc A_i and each replication k, define an indicator function $I_i(t, k)$:

$$I_i(t, k) = \Sigma_n \, \delta(t, t_i(n, k))$$

where the sum is taken over the total number of tokens passing over A_i in the course of replication k. The $\delta(\cdot, \cdot)$ function is the Dirac delta: a function which is zero almost everywhere and which integrates to unity when the value of one argument is included in the range of integration of the other.

The $I_i(t, k)$ are the most basic stochastic processes for a STAPN. However, more important is the counting process $J_i(t, k)$, which gives the number of tokens having traveled on A_i between time 0 and time t, in replication k. $J_i(t, k)$ is related to $I_i(t, k)$ by:

$$J_i(t, k) = \int_0^t I_i(s, k) \, ds$$

The processes $I_i(t, k)$ and $J_i(t, k)$ are the primitive stochastic processes from which the canonical measures will be constructed. Note that the algorithm for generating either of these two processes from observations of a STAPN's behavior is consistent with Assertion 4.

A.2.2 Forms of Measures

To see the differences between four forms of measures, consider J_i, the number of tokens having passed over arc A_i, as an example measure. This basic quantity can be interpreted as an instantaneous value; it may be averaged over a number of replications, over time, or both; the averages may be finite or infinite.

An instantaneous value of a measure is taken at one instant of time t in a single replication k. To make this dependence explicit, denote the instantaneous value of J_i as $J_i(t, k)$, as above. Since tokens are almost never on arcs, $J_i(t, k)$ is piecewise constant, with unit step discontinuities at a countable number of values of t and k.

An ensemble average is taken over several replications. Denoted by the form J_i (t), ensemble averages may be computed from the instantaneous values as:

$$J_i\ (t) = K^{-1}\ \Sigma_k\ J_i\ (t,\ k)$$

where K is the number of replications. Unless K is explicitly mentioned, assume K is infinite:

$$J_i\ (t) = \lim_{K\to\infty}\ \{K^{-1}\ \Sigma_k\ J_i\ (t,\ k)\}$$

A time average is, of course, taken over time. Time average measures are denoted by the form J_i (k), preserving the dependence on the replication index. Time averages may be computed from the instantaneous values as:

$$J_i\ (k) = T^{-1} \int_0^T J_i\ (t,\ k)$$

where T is the time interval of interest. Note that J_i (k) takes on the interpretation of a rate: the number of tokens which cross A_i per unit time. Again, unless T is explicitly specified, assume T is infinite:

$$J_i\ (k) = \lim_{T\to\infty}\ \{T^{-1} \int_0^T J_i\ (t,\ k)\}$$

If this limit exists, and the system does not exhibit periodic behavior, then J_i (k) is often refered to as a steady state measure.

Finally, a time-ensemble average is, as expected, taken over both time and replications. These measures are denoted by the form J_i, suppressing the dependence on time or replication index. These averages may be computed from any of the above in the obvious ways. Both averages are assumed infinite unless stated otherwise.

Which of these four forms is preferable? The answer depends on the uses to which they are to be put. Instantaneous values or ensemble averages are useful when the situation or system structure changes over time, since they are applicable at every instant of time. Time averages are useful when the system and its environment are in some sort of equilibrium, so that a steady state measure can be considered meaningful.

In subsequent subsections, only infinite ensemble averages and infinite ensemble-time averages (steady state measures) will be considered (e.g., J_i (t) and J_i). Definitions of canonical measures given in these forms can readily be adjusted to provide definitions of the measures in the other forms. However, if finite averages are used as measures, the limits of the averages (i.e., K or T) must be specified in order for the measures to be well defined.

A.2.3 Probabilities

The first canonical measures are associated with arcs leaving places. To each place corresponds a decision rule, which specifies by which arc each token departs. A certain fraction of the tokens that leave the place follow each of the exit arcs; these fractions are the canonical probability measures.

Formally, suppose place P_i has output arcs A_1, A_2, ..., AK. Define the K probability measures associated with those arcs as:

Given that a token leaves place i at time t in some replication, the probability that the token leaves on arc n is:

$$\rho_{in}(t) = I_n(t) \,/\, \Sigma_k I_k(t)$$

The steady state probability that a token leaves place i via arc n is:

$$\rho_{in} = \lim_{t \to \infty} \{J_n(t) \,/\, \Sigma_k J_k(t)\}$$

Since $\rho_{in}(t)$ is defined only at a countable number of points (since only a countable number of tokens are generated in a countable number of replications), care must be taken in the determination of the time average ρ_{in}. To keep ρ_{in} dimensionless, we take the definition of ρ_{in} as an assertion, not a direct consequence of the definition of $\rho_{in}(t)$.

All of the probabilities are unconditional probabilities: their definition assumes nothing is known about the activities in other parts of a net. Conditioning the probabilities on such knowledge can drastically change their values. For example, consider the partial net (Fig. A-1). Without knowledge

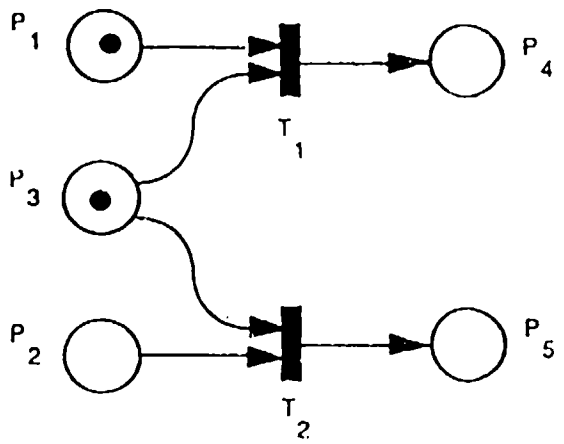


Figure A-1. Conditional Exit From a Place

of the placement of tokens in P_1 and P_2, ρ_{31} and ρ_{32} might both be, say, 0.5. However, conditioned on the marking of the net in Fig. A-1, the probability that the token in P_3 leaves through T_1 is 1.0. Thus a probability measure ρ_{ik} must be carefully interpreted; it is meaningful and well defined precisely when the only knowledge about the net is that some token is leaving place i.

Every place has at least one output arc, so at least one probability measure is defined for each place. If exactly one output arc leaves a place, the value of the probability measure for that arc will always be 1.0, irrespective of any other structure or activities of the net. This is a degenerate example of a canonical measure, and formally defined but rather useless in practice.

A.2.4 Rates

The second canonical measures are attached to transitions. Transitions fire; this is their essential activity. In most STAPNs, transitions fire repeatedly, and it is meaningful to measure how often they fire. The frequencies with which transitions fire are called the canonical rate measures.

Formally, suppose transition T_i has input arcs A_1, A_2, ..., A_K and output arcs A_{K+1}, A_{K+2}, ..., A_{K+L}. Due to the coordination role of transitions, and the fact that exactly one token passes over each and every one of these arcs when T_i fires, it is known that:

$$J_1(t) = J_2(t) = \ldots = J_{K+L}(t)$$

Without loss of generality, $J_1(t)$ can be used in the definition of the canonical rate measure for transition T_i:

The rate at which transition i fires is:

$$\lambda_i(t) = d/dt\ [\ J_1(t)\ ]$$

The steady state firing rate of transition i, if it exists, is:

$$\lambda_i = \lim_{T \to \infty} \{\ T^{-1}\ J_1(t)\ \}$$

The firing rate of a transition implies that tokens flow along each of its input arcs and output arcs at exactly the same rate. From these, token flow rates into and out of a place can be determined by summing arc flow rates over all input or output arcs of that place. Because of these relations, rate measures need not be defined for other elements of a STAPN.

A.2.5 Occupancies

The third canonical measures are connected with places. Places store tokens between their creation and destruction. The natural quantities to associate with places are the number of tokens which occupy those places; these are the canonical occupancy measures.

Formally, suppose place P_i has K input arcs labeled $A_1, A_2, \ldots, A_K$ and M output arcs $A_{K+1}, A_{K+2}, \ldots, A_{K+M}$. Tokens arrive in the place over any input arc, so the process which indicates the total number of arrivals into the place, by time t, is

$$\Sigma_k \, J_k \, (t)$$

Similarly, the process which indicates the total number of departures from the same place, by time t, is

$$\Sigma_m \, J_{K+m} \, (t)$$

Now, the number of tokens in the place at time t is some initial number of tokens specified by the marking of the STAPN, η_i (0), plus the number that have arrived, less the number that have departed:

The occupancy of place i is:

$$\eta_i \, (t) = \eta_i \, (0) + \Sigma_k \, J_k \, (t) - \Sigma_m \, J_{K+m} \, (t)$$

The steady state occupancy of place i, if it exists, is:

$$\eta_i = \eta_i \, (0) + \lim_{T \to \infty} \left\{ T^{-1} \int_0^T \Sigma_k \, J_k \, (t) - \Sigma_m \, J_{K+m} \, (t) \, dt \right\}$$

A.2.6 Delays

The fourth canonical measures apply to tokens. Tokens exist between their creation at one transition and their destruction at another. The natural quantities to associate with tokens are their lifetimes; these are the canonical delay measures.

Formally, consider all tokens created in place P_i and destroyed by output transition T_j. (All such tokens spend their lives in the same place. However, different timing models may describe the duration of their unavailable state, since they may have been created by different transitions.) Number these tokens in order of their creation, using the index n_{ij} to emphasize the dependence on both P_i and T_j. Let $t_i \, (n_{ij}, k)$ be the time of creation of the n_{ij}-th token to arrive at P_i in replication k; let $t_j \, (n_{ij}, k)$ be its time of

destruction at T_j. Note that delay measures are defined on a token by token basis, so we must replace time averages with token averages to form steady state measures:

The delay experienced, before time t, by tokens between arrival in place i and destruction at transition j is:

$$\tau_{ij}\ (n_{ij}) = t_j\ (n_{ij}) - t_i\ (n_{ij})$$

The steady state delay experienced by tokens between arrival at place i and destruction at transition j, if it exists, is:

$$\tau_{ij} = \lim_{N\to\infty} \{\ N^{-1}\ \Sigma_n\ \tau_{ij}\ (n)\ \}$$

Note that $t_{ij}\ (n_{ij}, k)$ is defined only when the n_{ij}-th token from T_i to T_j has been destroyed, that is, when $t_j\ (n_{ij}, k)$ is known. Tokens which have been created, but are still in a place, do not have a permanently defined lifetime and are thus excluded from this definition. Tokens which are part of the initial marking arrive at their respective places at time 0, when the STAPN starts to operate.

Finally, note that these delays are conditional delays: they represent the time a token spends in P_i given that it exits through T_j. They do not describe the delay which any token experiences in P_i, although the latter measure can be computed from the canonical delays and probabilities.

A.3 HORIZONTAL RELATIONS BETWEEN MEASURES

From the discussions above, it is clear that not all measures are independent. The most trivial example occurred in the case where a place has a single output arc: the probability for that arc is 1.0, regardless of anything else. This section identifies four classes of relations which exist between the canonical measures of a STAPN.

Consider the partial STAPN and its canonical measures in Fig. A-2. A number of relationships hold between the measures shown here. For example, equations which relate the steady state measures happen to be:

$\rho_{12} + \rho_{13} = 1.0$	Conservation of probability
$\lambda_1 = \lambda_2 + \lambda_3$	Conservation of rate
$\lambda_2 = \rho_{12} * \lambda_1$	Probability -- rate relation
$\lambda_3 = \rho_{13} * \lambda_1$	Probability -- rate relation
$n_1 = \lambda_1 * (\rho_{12} * \tau_{12} + \rho_{13} * \tau_{13})$	Occupancy -- rate -- delay relation

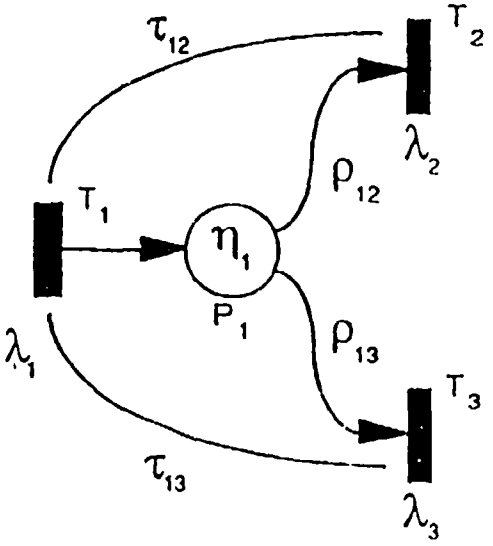


Figure A-2. Canonical Measures

The four types of relations exemplified above can be generalized; in fact, similar equations exist for the steady state measures surrounding every place in a STAPN. For measures expressed as ensemble averages, the analogous equations are not algebraic; they take the form of delay-differential equations (which can be viewed as relations between entire time functions, rather than between single values). Whatever the form, these equations provide the set of horizontal relations that apply within a single STAPN model.

A.3.1 Conservation of Probability

Recall that the canonical probabilities measure the fraction of tokens that leave a place along each of its exit arcs. Since every token that leaves a place must follow exactly one output arc, these fractions must sum to one.

Formally, suppose place P_i has output arcs $A_1, A_2, \ldots, A_K$. The first horizontal relation between measures is:

Conservation of Probability:
For ensemble averages:

$$\Sigma_k \rho_{ik}(t) = 1.0$$

For steady state statistics:

$$\Sigma_k \rho_{ik} = 1.0$$

These relations imply that one probability measure at every place is redundant--including the trivial situation where only one arc leaves a place.

A.3.2 Conservation of Rates

The second class of horizontal relations also follows from the fact that tokens do not evaporate from places. Every token that enters a place either stays there, or leaves to be destroyed at a transition. As long as tokens do not pile up in a place indefinitely, the sum of the output rates must approach the sum of the input rates.

Formally, suppose a place P_i has K input arcs labeled A_1, A_2, ..., A_K from transitions T_1, T_2, ..., T_K respectively, and M output arcs labeled A_{K+1}, A_{K+2}, ..., A_{K+M} to transitions T_{K+1}, T_{K+2}, ..., T_{K+M}. Recall the basic occupancy equation:

$$n_i(t) = n_i(0) + \Sigma_k J_k(t) - \Sigma_m J_{K+m}(t)$$

Differentiating both sides with respect to time, and substituting the definitions of rate statistics provides:

Conservation of Rates:
For ensemble averages:

$$d/dt\ n_i(t) = \Sigma_k \lambda_k(t) - \Sigma_m \lambda_{K+m}(t)$$

or

$$\Sigma_m \lambda_{K+m}(t) = \Sigma_k \lambda_k(t) - d/dt\ n_i(t)$$

For steady state statistics, if they and n_i exist:

$$\Sigma_m \lambda_{K+m} = \Sigma_k \lambda_k(t)$$

These relations imply that many rate statistics are redundant. The STAPN topology restricts the channels which tokens may follow; tokens that contribute to a rate statistic at the input to a place must eventually contribute to an output rate at the same place.

A.3.2 Probability -- Rate Relations

The third class of horizontal relations results from the process which determines how a token departs from a place. The total flow of tokens into a

place diverges into flows along each output arc, so that the total flow out of the place is the sum of the output transition firing rates. A probability describes the fraction of that flow which travels along each arc, and the flow along each arc equals the firing rate of the transition terminating that arc. Thus we can relate individual output transition firing rates to the total output firing rate.

Formally, again suppose a place P_i has M output arcs labeled A_1, A_2, ..., A_M to transitions T_1, T_2, ..., T_M respectively. From the definition of C_{in} (t):

$$I_n (t) = \Sigma_m I_m (t) * \rho_n (t)$$

Note that

$$d/dt\ J_n (t) = \lim_{K\to\infty} \{ K^{-1} \Sigma_k\ d/dt\ J_n (t, k) \}$$

so

$$\lambda_n (t) = \lim_{K\to\infty} \{ K^{-1} \Sigma_k\ I_m (t, k) \} = I_n (t)$$

Similar manipulations apply to the steady state statistics. From the definition of ρ_n:

$$\lim_{T\to\infty} \{ T^{-1} J_n (T) \} = \lim_{T\to\infty} \{ T^{-1} \Sigma_m J_m (T) \} * \rho_n$$

and passing to the limit:

$$\lambda_n = \{ \Sigma_m \lambda_{in} \} * \rho_n$$

These arguments give us the third kind of horizontal relation:

Probability-Rate Relations:
For ensemble averages:

$$\lambda_n (t) = \Sigma_m \lambda_m (t) * \rho_n (t)$$

For steady state statistics, if they exist:

$$\lambda_n = \Sigma_m \lambda_{in} * \rho_n$$

These relations can be used two ways. If canonical probabilities are known, then all rates for the output transitions of a place can be deduced from any one such firing rate. If all rates are known, then we can solve for all probabilities without any other information.

A.3.4 Occupancy-Rate-Delay Relations

The final class of horizontal relations exploit two different interpretations of the same quantity: the number of token-seconds spent in a place. This can be computed by adding up the lifetimes of all tokens that transit a place, or by integrating the number of tokens present over time.

Formally, suppose a place P_i has K input arcs labeled $A_1, A_2, \ldots, A_K$ from transitions $T_1, T_2, \ldots, T_K$ respectively, and M output arcs labeled $A_{K+1}, A_{K+2}, \ldots, A_{K+M}$ to transitions $T_{K+1}, T_{K+2}, \ldots, T_{K+M}$. Consider the nth token that passes through Pi, departing through T_{K+m} at time

$$t = t_{K+m}\ (n)$$

At this time, the total number of tokens having passed through P_i is

$$\Sigma_m\ J_{K+m}\ (t)$$

Assuming first-in, first-out passage of tokens through P_i, this n-th token will have arrived at Pi at some time $t - \tau_i$ which satisfies:

$$\Sigma_k\ J_k\ (t - \tau_i) = \Sigma_m\ J_{K+m}\ (t)$$

Differentiating both sides with respect to t(n),

$$\Sigma_k\ \lambda_k\ (t - \tau_i) * (1 - d/dt\ \tau_i) = \Sigma_m\ \lambda_{K+m}\ (t)$$

Rearranging terms, the delay encountered by a token leaving P_i at time t is τ_i, given by the delay-differential equation:

$$d/dt\ \tau_i = \{\ \Sigma_k\ \lambda_k\ (t - \tau_i) - \Sigma_m\ \lambda_{K+m}\ (t)\ \}\ /\ \Sigma_k\ \lambda_k\ (t - \tau_i)$$

Now consider the steady state statistics. The total amount of time spent by those tokens in P_i, which have left via T_{K+m} by time t, is:

$$\theta_{i,K+m}(t) = S_n\, t_{i,K+m}(n, t)$$

The total amount of time spent by all tokens in P_i which have left by time t is:

$$\theta_i(t) = \Sigma_m\, \theta_{i,K+m}(t)$$

The number of tokens departing P_i by time t is:

$$\theta_m\, J_{K+m}(t)$$

so the average amount of time, per token, spent in P_i is:

$$\theta_i(t) \;/\; \Sigma_m\, J_{K+m}(t)$$

Now, an alternative way to compute $\theta_i(t)$ is:

$$\theta_i(t) = \int_0^t n_i(t)$$

Letting τ_i be the average time any token spends in P_i, the combination of the last two forms yields:

$$\tau_i = \{\, T^{-1} \int_0^T n_i(T) \,\} \;/\; \{\, T^{-1}\, \Sigma_m\, J_{K+m}(T) \,\}$$

Passing to the limit in T gives Little's formula, well known from queuing theory:

$$\tau_i = n_i \;/\; \Sigma_m\, \lambda_{K+m}$$

These arguments give us:

Occupancy -- Rate -- Delay Relations:
For ensemble averages:

$$d/dt\; \tau_i = \{\, \Sigma_k\, \lambda_k(t - \tau_i) - \Sigma_m\, \lambda_{K+m}(t) \,\} \;/\; \Sigma_k\, \lambda_k(t - \tau_i)$$

For steady state statistics, if they exist:

$$n_i = \tau_i * \Sigma_m \lambda_{K+m}$$

These equations are not quite complete, since they involve the unconditional delay in P_i, namely τ_t. Instead of applying just to tokens leaving P_i by a specific arc, τ_i is an average over all tokens passing through P_i, and has not been seen before. Conditional delays, τ_{ij}, have been defined previously, but are defined with respect to a specific transition T_j. Fortunately, the two delays are easily related to one another. For ensemble averages, if the $n_{i,K+m}$-th token departs P_i via T_{K+m} at time t,

$$\tau_i\ (t) = \tau_{i,K+m}\ (n_{i,K+m})$$

For steady state statistics, the unconditional delay is simply a weighted average of the conditional delays:

$$\tau_i = \Sigma_m \rho_{i,K+m} * \tau_{i,K+m}$$

A.3.5 Implications for Minimality

Clearly the entire set of canonical measures for a STAPN, while entirely meaningful, includes redundant elements. The four types of horizontal relationships described above apply to any STAPN, and thus can be used to eliminate redundant measures. By first identifying all canonical measures, then finding all of these horizontal relationships, one has a simple procedure for eliminating the redundant measures. Simply select one relationship, eliminate one measure which appears in it, use it to replace all occurrences of the eliminated measure in all other relations, then repeat for each other relation. Of course, the equations above provide no guidance pertaining to the order in which relationships and measures should be selected in this process.

The power of these relations can be clearly seen when steady state statistics are used. For example, one can use the occupancy-rate-delay relations to compute all occupancies from rates and delays, so all occupancy measures can be eliminated. Then one can use the probability-rate relations to compute all probabilities from the rates, so all probabilities can be discarded. Finally, the conservation of rate relations eliminate a rate measure from one output transition of each place. By following these steps, a majority of the canonical measures can be removed from the original set of candidate measures.

There is no guarantee that the canonical relations are the only sources of dependence among measures. Indeed, when values of measures are to be computed, additional relations must be made available in order for the values to

be completely determined. The four classes of relations presented here are, however, the only relations which always hold and which may be determined solely from the topology of the STAPN.

A.4 REFINEMENT OF MODELS

Petri net models of real systems are notoriously complex -- often prohibitively so. In order to manage the unavoidable complexity of C^3 systems, we have suggested that STAPN models could be arranged in hierarchies, with each level representing the same system but with differing amounts of detail. Well defined connections between models at successive levels allow the general structure depicted in a higher level model to be transferred to lower level models. In addition, these same connections can help structure the model building process, and give rise to quantitative relations between measures at neighboring levels.

A.4.1 Disaggregation and Enhancement

Two mechanisms exist for expanding a model from one level of detail to another, based on opposing philosophies. The first is strict structured decomposition or disaggregation, where each element of a level N model decomposes into a set of elements at level N+1, and where the level N+1 model contains nothing but these decomposed elements. We naturally think of disaggregation in terms of geographical regions, where nations disaggregate into nonoverlapping territories, and every acre of land is contained in some territory. Similarly, organizations often disaggregate from divisions to groups to people, and every person is a member of one group, every group is part of one division. Strict disaggregation is a powerful technique when it can be used, as it implies precise relationships (homomorphisms) between the structural elements depicted in successive model levels.

Unfortunately, very few real systems can be disaggregated so that their higher levels are just aggregated versions of their lower levels. More often, the lower levels contain details that have no counterpart at the higher levels. Yes, the United States can be disaggregated into fifty states and several territories, but where do national parklands and Indian reservations fit into this scheme? Yes, organizations can be disaggregated into divisions, but where do interdivisional programs and matrix management fit in?

To deal with real C^3 systems, we must be prepared to consider details relevant to level N+1 which do not appear at all in higher level models. For example, special purpose messages, which are part of a protocol to manage a communications network, should be suppressed in any simple model of the mission which that network supports. Such details which appear afresh at level N+1, without being modeled at level N, are enhancements to the model of level N.

The distinction between disaggregation of a model, which implies strong relations between models at two adjacent levels, and enhancement of that model, which precludes any such relations, becomes particularly important

when we derive vertical relations between the measures associated with the two levels. Subsection A.4 will revisit this distinction; for now, we simply explore three techniques for increasing the detail of an existing model: two disaggregation methods (one for transitions and another for places) and enhancement.

A.4.2 Disaggregation at Transitions

The simplest disaggregation technique expands one transition at level N into several transitions at level N+1. This partitions the events (physical boundary crossings) modeled by the higher level transition into subsets of events, one subset for each lower level transition.

Formally, consider two STAPN models of the same system, organized at two levels of detail. The model at level N contains some transition T_A, and the model at level N+1 has some transitions $T_{B,1}$, $T_{B,2}$, ... $T_{B,M}$. Define:

> Transitions $T_{B,1}$, $T_{B,2}$, ... $T_{B,M}$ form a strict disaggregation of transition T_A if and only if there is a one-to-one correspondence between firings of the former and firings of the latter.

A simple illustration of disaggregation at a transition is shown in Fig. A-3. Here, transitions T_{B2} and T_{B3} form a disaggregation of transition T_{A2}. (To be precise, this is the case if and only if T_{B1} is a trivial disaggregation of T_{A1}, so every token entering P_{A1} corresponds to a token entering P_{B1}.) Since tokens may leave P_{A1} only via T_{A2}, and P_{B1} only via T_{B2} or T_{B3}, there is a one-to-one correspondence between firings of the shaded transitions in the two models.

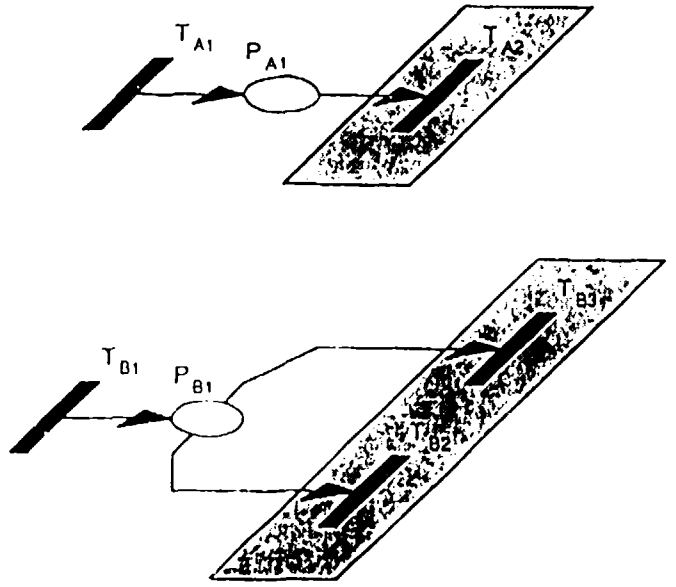


Figure A-3. Disaggregation at a Transition

Physically, disaggregation at transitions divides the flow of objects or messages across some boundary into component flows. The components may be defined in two ways. First, a single class of objects at the higher level may be divided into several classes of objects at the lower level. For example, messages arriving at a facility may all be lumped together at the higher level, but broken out by type at the lower level. Secondly, the actual boundary may be refined, being broken into segments across which objects of the same type flow. For example, the flow of enemy helicopters across the forward edge of the battle area can be decomposed by the sector from which they arrive.

In summary, disaggregation at transitions supports hierarchical decomposition of token classifications and of boundaries.

A.4.3 Disaggregation at Places

The second disaggregation technique expands one place at level N into several places and transitions at level N+1. This refines the model of processing done at a place in the higher level model into individual processing steps.

Formally, consider two STAPN models of the same system, organized at two levels of detail. The model at level N contains some place P_A, and the model at level N+1 has some places $P_{B,1}$, $P_{B,2}$, ... $P_{B,M}$. Define:

> Places $P_{B,1}$, $P_{B,2}$, ... $P_{B,M}$ form a strict disaggregation of place P_A if and only if there is a one-to-one correspondence between tokens in the former and in the latter.

A simple illustration of disaggregation at a place is shown in Fig. A-4. Here, places P_{C1}, P_{C2} and P_{C3} form a disaggregation of place P_{B1}. (Again, this is the case if and only if each $T_{C,m}$ is a trivial disaggregation of transitions $T_{B,m}$, m = 1, 2, and 3, so every token entering P_{C1} corresponds to a token entering P_{B1}, and every token leaving P_{C2} or P_{C3} corresponds to a token leaving P_{B1}.) Note that the two processing paths in the original model (T_{B1} to P_{B1} to T_{B2}, and T_{B1} to P_{B1} to T_{B3}) are each decomposed into a single sequence of transitions and places in the lower model, but that the latter share T_{C1} and P_{C1}. Note also that the new transitions, T_{C4} and T_{C5}, have no counterparts in the original model, so they are not disaggregations of any original transitions. If the initial number of tokens in P_{C1}, P_{C2} and P_{C3} equals the initial number of tokens in P_{B1}, then there will always be a one-to-one correspondence between tokens in the shaded places of the two models.

Physically, disaggregation at places divides the processing of objects or messages in a region or facility into processing substeps. For each pair of input and output transitions in the original model, a sequence (more generally, an acyclic network) of places and transitions appears in the lower model to convey details of the original processing. Just as tokens in the original model share residence in the original place, so the refined processing pathways may share transitions and places. However, no coordination mechanism other than simple sequencing along each pathway is possible using place disaggregation, as more sophisticated coordination mechanisms require additional tokens, usually to represent memory in the coordination mechanism, which have no counterparts in the original model.

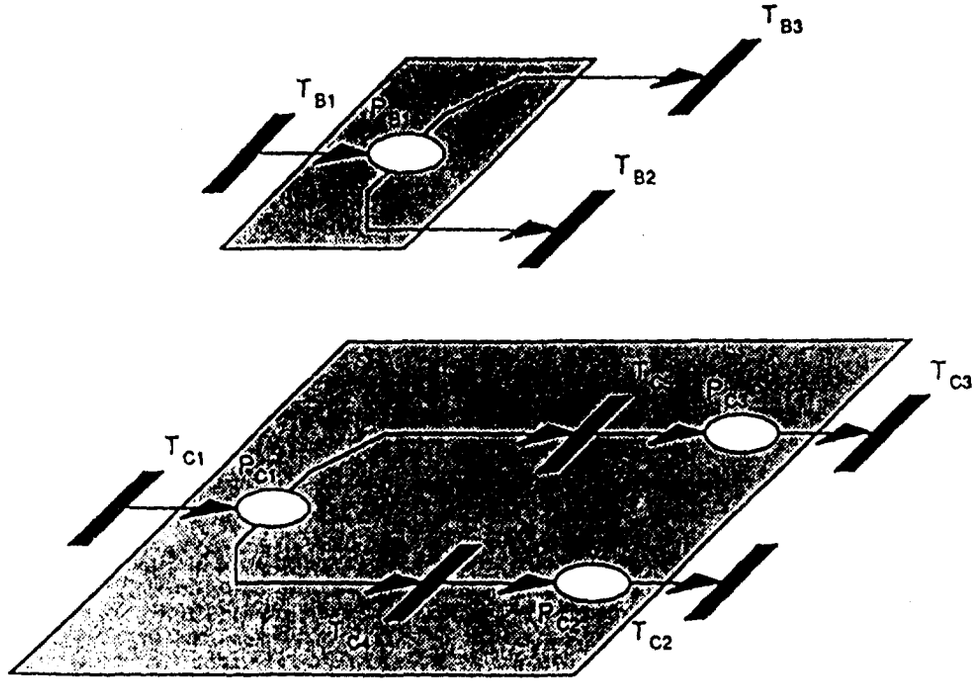


Figure A-4. Disaggregation at a Place

The natural physical candidates for place disaggregation are decomposition of regions into subregions, division of facilities into components, and segmentation of procedures into steps. For examples, a high level model might represent an entire nation with one place, using transitions between places to model border crossing events; a lower level model might use a place for each state or county, with additional transitions modeling internal border crossings. The North Cheyenne Mountain Complex might be a single place in one model; each operations center might be a place in a lower level model, with some additional transitions modeling message or personnel movement between centers. The launch preparation process for a strategic missile might be captured by one place in a high level model; at a lower level, individual places might represent separate preparatory processes such as fueling, arming, and targeting.

In summary, disaggregation at places supports hierarchical decomposition of regions, facilities, and procedures.

A.4.4 Simultaneous Disaggregation/Subnetworks

Parenthetical comments following Figs. A-3 and A-4 indicated that simultaneous disaggregation at both transitions and places is needed in order to assert that either one is valid. That is, one-to-one correspondences between tokens can be maintained if and only if there exist one-to-one correspondences between the events that create and destroy tokens (transition firings). Thus procedures to validate that a place disaggregation is correct must introduce compatible transition disaggregations, and vice versa.

Formally, the correspondences between tokens and transition firings provided by a simultaneous disaggregation provide a well defined homomorphism between two model levels. In subsection A.4, we will see that this homomorphism creates exact relationships between measures at the two levels.

From a practical perspective, simultaneous disaggregation is more readily understandable than artificially enforced separations between disaggregation at transitions and at places. In fact, simultaneous disaggregation can be interpreted quite naturally as subnetwork expansion. Each place is disaggregated into a subSTAPN, which may model the decision rule or timing model for the original place in more detail. Each transition is disaggregated to ensure that the boundaries of the subSTAPNs are compatible: the output transitions for one subSTAPN must match the input transitions of any subSTAPNs that follow it. Thus simultaneous disaggregation permits construction of hierarchies of models which have precise correspondences between elements at successive levels.

A.4.5 Enhancement

However, simultaneous disaggregation is not sufficient to model real C^3 systems. The high level structure of these systems only becomes apparent when some details are completely suppressed. We must have a method for resurrecting these details as the model levels become more detailed, or risk losing the completeness property of the models. That method is enhancement: adding additional STAPN primitives to a model for which no corresponding element exists at the level above.

There is no formal definition of enhancement, but the concept is adequately illustrated by example. In Fig. A-4, we disaggregated a place into three places and two transitions, but added no information about the decision rule for the original place. Suppose that logic is to send alternating tokens to P_{C2} and P_{C3}. The lower level model of Fig. A-4 can be enhanced to capture this special coordination mechanism as shown (Fig. A-5). Here, tokens alternately occupy P_{D4} and P_{D5}, thus enabling transition T_{D4} for every other token that becomes available in P_{D1}. However, there is no token at the higher level that corresponds to the token initially in P_{D5}. (Similarly, no transitions fire at the higher level when either T_{D4} or T_{D5} fire, as seen in Fig. A-4.) Thus P_{D4} and P_{D5} constitute an enhancement to the higher level model.

Physically, enhancement adds structure to a model. It does not nullify any aspects of a higher level model, but introduces new objects and coordination mechanisms to extend the detail of that model. For most simple STAPN models, complexity not represented by the network itself is hidden in the decision rules and timing models. Therefore, we would expect enhancement to inject more visibility into these elements as a model is refined.

Figure A-5 is a simple example of how an allocation decision rule, modeled simply as the separation of two token flows in Fig. A-3, can be made explicit. Similar mechanisms express the detail of other common allocation rules, such as allocating a message to an operator with the shortest queue. Timing models

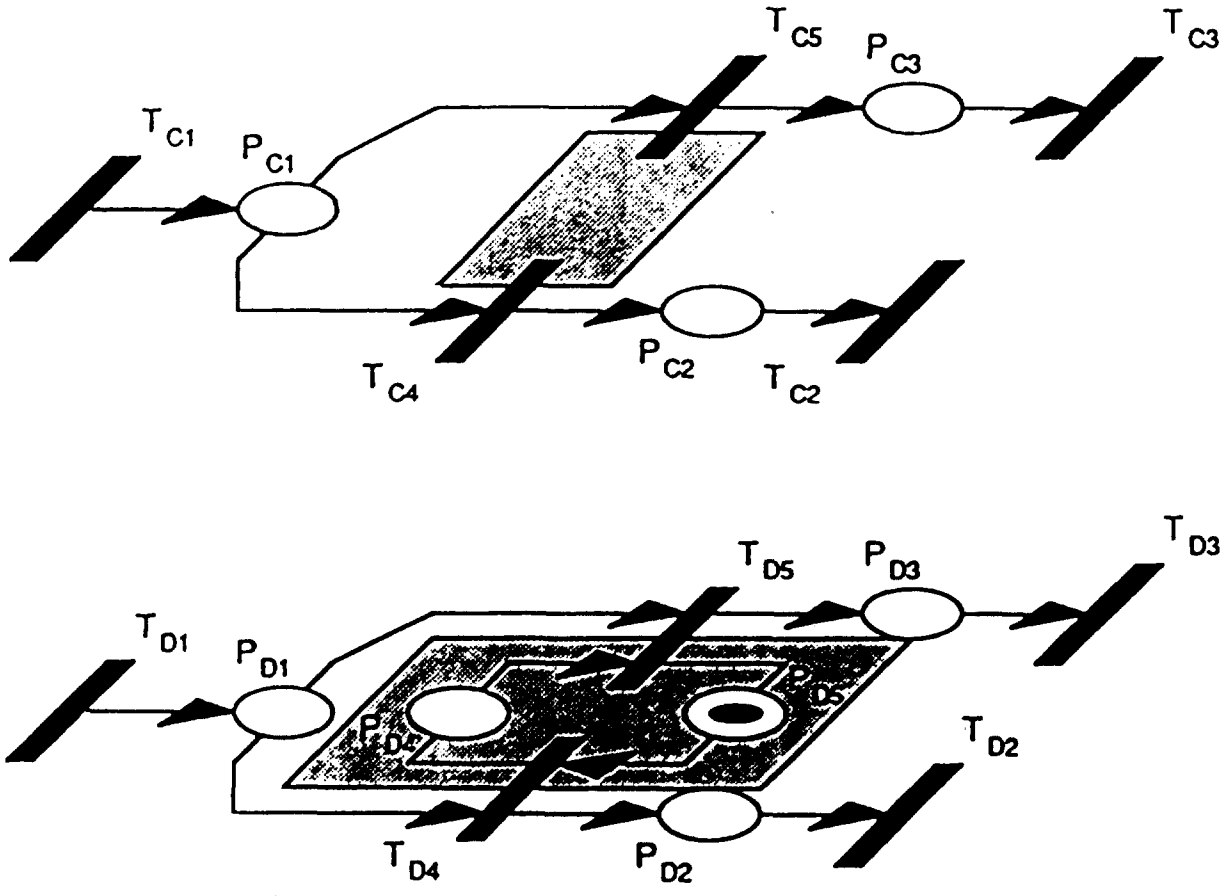


Figure A-5. Enhancement

used in higher level models usually represent activities of long duration, such as establishing the identity of an object, and a lower level enhancement would make explicit the time taken when false identities are conjectured and rejected, perhaps several times, before the true identity is established.

Finally, we saw that disaggregation only permits processes to be decomposed into sequential subprocesses. In order to introduce the parallelism and related coordination mechanisms so vital to C^3 systems, we cannot rely on disaggregation alone. Enhancement provides the opportunity to introduce parallel processing paths and additional coordination between them, as in Fig. A-5.

In summary, enhancement supports hierarchical decomposition of decision rules, timing models, and coordination mechanisms.

A.4.6 Implications for Completeness

In addition to the usual issues of model fidelity, internal completeness is a property of hierarchies of models that can potentially be verified.

The disaggregation techniques provide the structure for validating internal completeness.

In what sense can one expect a hierarchy of models to be internally complete? It has been repeatedly argued that much detail present in lower levels must be suppressed from higher levels for the models to be managable. Enhancement is a necessary technique for building hierarchical models of C^3, so one cannot ask that all features present in one level be present in all levels above. One can, however, demand that nothing be lost as we work more detail into the models -- i.e., that every object and event captured by a high level model be included in some component of each lower level model.

In STAPN terminology, internal completeness demands that every token created at level N have a unique counterpart at level N+1, N+2, ..., and that every transition firing at level N corresponds to a unique transition firing at every level below it. Suppose that:

> Assertion A-1: Every STAPN model of a system can be decomposed by successive disaggregations (at either places or transitions) and enhancements.

Now, the properties required for strict disaggregation (one-to-one correspondences between tokens and transition firings at successive levels of the model) are transitive (so that they apply between levels N and N+K+M if they apply between levels N and N+K, and between levels N+K and N+M). Thus if successive disaggregation-enhancement steps are used to build a hierarchy of models in a top-down fashion, any token or firing in level N will have a unique counterpart at every level below, so

> Fact A-1: Every STAPN model constructed from top-down iterations over disaggregation-enhancement steps, where the disaggregation is performed at every place and every transition, is internally complete.

Thus one can ensure internal completeness for a collection of STAPN models by carefully structuring the model-building process, so that disaggregation and enhancement are alternated as the model is built in a top-down fashion.

A.5 VERTICAL RELATIONS BETWEEN MEASURES: AGGREGATION

Suppose one has two versions of a model at different levels of detail, and they are related to one another as described in subsection A.4. How do the connections between models carry over into relationships among measures?

For each of the four types of canonical measure, the disaggregation methods produce a set of vertical relations which determine values for the higher level measures from lower level values. If the conditions of Fact A-1 are followed, one vertical relation is produced for each high level measure. Since the structures introduced by enhancement are not part of the disaggregations, the measures associated with the enhancement elements will not appear in any of the vertical relations for the level above.

A.5.1 Paths and Path Sets

We will use the same notation in the discussion of all four classes of vertical measures. Any simultaneous disaggregation process starts with some place at level A, P_A, its input transitions $T_{A,1}$, $T_{A,2}$, ... $T_{A,K}$, and its output transitions $T_{A,K+1}$, $T_{A,K+2}$, ..., $T_{A,K+M}$. Disaggregation at the input transitions produces transitions $T_{B,1}$, $T_{B,2}$, ... $T_{B,L}$ at the next level B. Disaggregation at the output transitions produces transitions $T_{B,L+1}$, $T_{B,L+2}$, ... $T_{B,L+N}$. Finally, disaggregation of PA,i produces several new places $P_{B,1}$, $P_{B,2}$, ... $P_{B,J}$.

Common to several definitions of vertical relations is the notion of a path set. A path under P_A is an alternating sequence of transitions and places which (a) starts at one of the disaggregated input transitions $T_{B,1}$, $T_{B,2}$, ... $T_{B,L}$, (b) terminates at one of the disaggregated output transitions $T_{B,L+1}$, $T_{B,L+2}$, ... $T_{B,L+N}$, (c) only contains a subset of $P_{B,1}$, $P_{B,2}$, ... $P_{B,J}$, and (d) is acyclic. The collection of all such paths is the path set for P_A, denoted as S_A. The collection of all paths in S_A that terminate on transitions diaggregated from T_{Ai} is denoted by the set $S_{A,Ai}$.

This definition of paths, along with the properties of the disaggregations which produce them, results in an essential characteristic of a path set: If several paths connect any one disaggregated input transition to any disaggregated output transition, they diverge and converge only at places.

The proof of this fact is by contradiction. Suppose two distinct paths in a path set diverge at a transition. Suppose this transition, common to both paths, fires. This creates one token in each of the two next places of each path. By the definition of place disaggregation, each of these two tokens must bear a one-to-one correspondence with tokens in the original, aggregated place. Since only one token is created in the aggregated place in the upper level model by this firing, these two tokens are in a one-to-one correspondence with the same token, thus producing a contradiction. The proof for convergence at places is proven by the same arguments, working backwards in time from the disaggregated output transition.

There may be several paths containing any single place. For example, consider the higher level system in Fig. A-3, which we refined into the lower level system of Fig. A-5. These two models are shown together in Fig. A-6. In this diagram, the original place P_{A1} has a path set S_{A1} containing two paths:

$$T_{D1} \rightarrow P_{D1} \rightarrow T_{D4} \rightarrow P_{D2} \rightarrow T_{D2}$$

$$T_{D1} \rightarrow P_{D1} \rightarrow T_{D5} \rightarrow P_{D3} \rightarrow T_{D3}$$

Neither P_{D4} nor P_{D5} appear in any path set, since they resulted from enhancement, not disaggregation at P_{A1}. For this reason, the two acyclic sequences:

$$T_{D1} \rightarrow P_{D1} \rightarrow T_{D4} \rightarrow P_{D4} \rightarrow T_{D5} \rightarrow P_{D3} \rightarrow T_{D3}$$

$$T_{D1} \rightarrow P_{D1} \rightarrow T_{D5} \rightarrow P_{D5} \rightarrow T_{D4} \rightarrow P_{D2} \rightarrow T_{D2}$$

are not in the path set for P_{A1}.

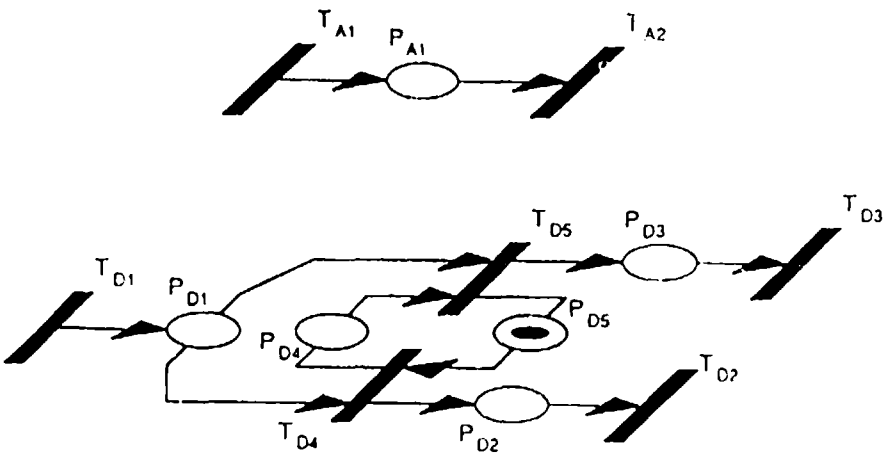


Figure A-6. Example for Vertical Relations

Continuing the example shown in Fig. A-6, five vertical relations hold between the canonical measures for the two systems. The equations which relate the steady state measures of the two models are:

$$\rho_{A1,A2} = \rho_{D1,D4} + \rho_{D1,D5} \qquad \text{Aggregation of probability}$$

$$\lambda_{A1} = \lambda_{D1} \qquad \text{Aggregation of rate}$$

$$\lambda_{A2} = \lambda_{D2} + \lambda_{D3} \qquad \text{Aggregation of rate}$$

$$\eta_{A1} = \eta_{D1} + \eta_{D2} + \eta_{D3} \qquad \text{Aggregation of occupancy}$$

$$\tau_{A1,A2} = \rho_{D1,D4} * \{\tau_{D1,D4} + \tau_{D2,D2}\} + \rho_{D1,D5} * \{\tau_{D1,D5} + \tau_{D3,D3}\} \qquad \text{Aggregation of delay}$$

The next four subsections derive general versions for each of these four classes of canonical vertical relations.

A.5.2 Aggregation of Probability

The canonical probabilities measure the fraction of tokens that leave a place along each of its exit arcs. As places are concatenated along a path,

these fractions multiply, as the flow is subdivided at each place. As paths converge, these fractions add. The total fraction of tokens that reach each disaggregated output transition is the sum of the fractions for each path terminating at that transition.

Formally, one must begin by defining some intermediate variables for the paths. The weight of a path s under P_{Ai}, denoted w_s, is a normalization factor times the product of the probabilities for each {place--output transition} pair encountered along the path. The normalization factor is the fraction of tokens created by the initial transition on the path. If a path begins at transition $T_{B,i}$, then the normalization factor is:

$$f_{B,i} = \lambda_{B,i} / \Sigma_l \lambda_{B,l}$$

These statements stand as is for steady state statistics; for an ensemble average $w_s(t)$, the normalization factor and probabilities to be multiplied together must be taken from the times at which each transition fired. That is, if the path s is:

$$T_{B1} \rightarrow P_{B2} \rightarrow T_{B3} \rightarrow P_{B4} \rightarrow T_{B5}$$

then the weight of the path is:

$$w_s(t) = \rho_{B4,B5}(t) * \rho_{B2,B3}(t - \tau_{B4,B5}) * f_{B1}(t - \tau_{B4,B5} - \tau_{B2,B3})$$

where

$$f_{B,i}(t) = g_{B,i}(t) / S_l \lambda_{B,l}(t)$$

Thus each probability measure is evaluated increasingly far in the past, as delays accumulate before the final transition fires. For steady state statistics,

$$w_s = \rho_{B4,B5} * \rho_{B2,B3} * f_{B1}$$

as all dependence on time is eliminated.

Now we can state the first vertical relation between measures. For each P_A, and each of its output transitions T_{Ai}:

Aggregation of Probability:
For ensemble averages:

$$\lambda_{A,Ai}(t) = \Sigma_s w_s(t)$$

For steady state statistics:

$$\rho_{A,Ai} = \Sigma_s w_s$$

where the sums are taken over paths s in $S_{A,Ai}$.

A.5.3 Aggregation of Rate

The canonical rates measure the frequency with which transitions fire. Disaggregation at a transition maintains a one-to-one correspondence between firings of the original transition and those of its components in the lower level model. Thus the firing rate for a transition at one level is the sum of the firing rates of the transitions into which it has been disaggregated.

Formally, a vertical relation can be defined for every transition in the original model. Let that transition be $T_{A,i}$; let it be disaggregated into N transitions $T_{B,1}$, $T_{B,2}$, ..., $T_{B,N}$. The second vertical relation between measures is:

Aggregation of Rate:
For ensemble averages:

$$\lambda_{A,i}(t) = \Sigma_n \lambda_{B,n}(t)$$

For steady state statistics:

$$\lambda_{A,i} = \Sigma_n \lambda_{B,n}$$

A.5.4 Aggregation of Occupancy

The canonical occupancies measure the number of tokens in places. Disaggregation at a place maintains a one-to-one correspondence between tokens in the original place and those in its components in the lower level model. Thus the occupancy of a place at one level is the sum of the occupancies of the places into which it has been disaggregated.

Formally, a vertical relation can be defined for every place in the original model. Let that place be $P_{A,i}$; let it be disaggregated into J places $P_{B,1}$, $P_{B,2}$, ..., $P_{B,J}$. The third type of vertical relation between measures is:

Aggregation of Occupancy:
For ensemble averages:

$$\lambda_{A,i}(t) = \Sigma_j \lambda_{B,j}(t)$$

For steady state statistics:

$$\lambda_{A,i} = \Sigma_j \lambda_{B,j}$$

A.5.5 Aggregation of Delay

The canonical delays measure the time between the creation of a token and its destruction. When output transitions are disaggregated, the set of tokens over which ensemble averages are taken may be partitioned. When a place is disaggregated, the life of one token may be broken into segments, represented by a sequence of tokens in the disaggregated model. The place may also be disaggregated into a set of alternative paths. One can reconstruct the lifetime of an original token by averaging the sum of token's lives along each disaggregated path, weighting the terms of that average by the fraction of tokens following each path.

Formally, another set of intermediate variables must be defined for the paths. The length of a path s under P_{Ai}, denoted l_s, is the sum of the delays for each {place--output transition} pair encountered along the path. As with weights on paths, this statement stands as is for steady state statistics; for an ensemble average $l_s(t)$, the delays to be summed must be taken from the times at which each token leaves each place. If the path s is:

$$T_{B1} \rightarrow P_{B2} \rightarrow T_{B3} \rightarrow P_{B4} \rightarrow T_{B5} \rightarrow P_{B6} \rightarrow T_{B7}$$

then the length of the path is:

$$l_s(t) = \tau_{B6,B7}(t) + \tau_{B4,B5}(t - \tau_{B6,B7}) + \tau_{B2,B3}(t - \tau_{B6,B7} - \tau_{B4,B5})$$

Now the fourth vertical relation between measures can be stated. For each P_A, and each of its output transitions T_{Ai}:

Aggregation of Delay:
For ensemble averages:

$$\tau_{A,Ai}(t) = \Sigma_s w_s(t) * l_s(t)$$

For steady state statistics:

$$\tau_{A,Ai} = \Sigma_s w_s * l_s$$

where the sums are taken over paths s in $S_{A,Ai}$.

A.5.6 Implications for Minimality

If the prescription for Fact A-1 to hold is followed, it will be ensured that every transition and place at level N is disaggregated, perhaps trivially, into elements at level N+1. If this is the case, it has been shown that a vertical relation can be constructed for every level N measure. Thus knowledge of the values for all level N+1 measures completely determines values for all level N measures. Thus the level N measures are redundant. If one seeks sets of measures that are independent, then one can only take measures from one level at a time.

Therefore, the ultimate set of measures desired are those from one level of a hierarchical STAPN model. Since there is one set for each level of the model, nested sets of measures have indeed been constructed, where higher level measures are simple aggregations of lower level measures.

A.6 EVALUATION OF MEASURES

Now all of the basic concepts required to systematically generate sets of measures for Command and Control systems are in place. For completeness, a review of some techniques for computing values for those measures is in order. Although evaluation of measures is outside the scope of this work, numerous comments on evaluation have appeared throughout, and some techniques merit a brief review.

Evaluation techniques fall into three classes. In increasing order of expense, these are analysis techniques, where values emerge as the solution to a set of equations; simulation, where the behavior of the system is mimicked by a combination of hardware and software models; and experiments, where the real system is operated in a manner representative of actual wartime conditions. This section cannot review all three techniques in depth, and so focuses on those aspects which are particularly germane to STAPN models.

The major contribution of the STAPN framework is not a set of new evaluation techniques; those listed below are commonly used and well understood. Rather, it establishes a common intellectual framework which facilitates comparisons between data obtained from different evaluations. Thus, for a single system, a numerical analysis, simulation results, and experimental data can be directly compared and checked for consistency.

The major question still to be addressed is the construction of enough additional relations between measures to permit a single solution for their values to emerge. Since the canonical horizontal and vertical relations arise from the structure of the network alone, the obvious sources for the additional relations are decision rules, timing models, and attribute maps. For example, timing models may relate delays to arrival rates and occupancies, and decision rules may produce models which determine probabilities from delays, occupancies, and rates.

A.6.1 Analysis

The most highly developed analytic techniques are those from critical path analysis and queuing theory. Since both PERT charts and queuing theoretic models are very similar to STAPN models, although more limited in scope, these techniques can at least be applied to a subset of STAPN models.

Generally, queuing analysis computes steady state values for measures. Additional relations to generate a solution are derived from analyses of standard queuing structures, such as single-server, first-in-first-out queues with reneging from the queue. The equations constructed by these analyses provide relations between rates, delays, and occupancies, as desired. Unfortunately, the equations are strictly correct only when a number of restrictive, and often unrealistic, assumptions are imposed for the probability distributions of various measures (e.g., the interval between transition firings is an exponentially distributed random variable).

Less work has been done on analytic evaluation of ensemble averages. These offer the potential of describing changes in behavior of a system as time progresses, as well as evaluating systems without any well defined steady state. At best, current technology permits differential equations to be derived for specific ensemble averages, such as the occupancy of a place representing a queue, in rather simple systems with many simplifying assumptions.

A.6.2 Simulation

Simulation is by far the most popular method for evaluating measures. Simulation has a reputation for providing visibility into the relations between reality and the assumptions built into a model intended to mimic that reality. This reputation is well deserved, except in cases where complexity overwhelms an analyst and visibility gives way to obscurity.

STAPN models are quite well suited for simulation. Indeed, a number of software products have appeared, particularly in Europe, which are tailored to simulation of some form of Petri net. However, none support all features necessary to directly simulate the STAPN models described in subsection 2.2. (Of course, such models can be simulated indirectly by translating them into the constructs of existing special purpose simulation languages).

The appeal of STAPN simulation lies in the fact that the basic events are transition firings, and these take place at discrete points in time. The marking of a STAPN changes only at a firing, so data structures need to be updated only at these times. Thus STAPNs are ideal candidates for discrete event simulation techniques, where computational load is determined by the number of events which occur in a system, not some arbitrarily chosen integration step size.

In addition, the hierarchical structure of the STAPN models described here simplifies the collection of output statistics. As each transition at

the lowest model level fires, several measures must be adjusted: the rate for that transition, the occupancies of its input and output places, and the probabilities and delays for arcs leading from input places to the transition. The vertical relations determine exactly how to propagate these changes up the hierarchy of measures, so that measures at all levels of detail can be accessed while the simulation is in progress.

However, simulations (including STAPNs) can be expensive in two ways. First, to construct a large simulation of a real system is a major software engineering effort. The availability of general purpose tools to aid this process has alleviated, but not eliminated, the large amount of manpower required to build a realistic simulation in which all assumptions are consistent. Modifying large simulations is also notoriously difficult.

The second expense associated with simulation is computer time. While one replication of a simulation may be rather inexpensive, an extremely large number of replications are usually needed for statistically significant ensemble averages to be computed.

A.6.3 Exercise and Experiments

The most expensive method for evaluating statistics is also the most realistic -- run the real system and see how it behaves. The major drawback to experiments or exercises is the significance of the results, as very few replications can be run to give statistical significance to ensemble averages.

The major potential contribution of the STAPN framework to evaluation by experiment is in the data collection and reduction processes. Prior to an exercise, data collectors must be instructed to observe and report specific events. The precision of a well built STAPN model can help define exactly what constitutes an event -- exactly what object is to cross what boundary as the event takes place. In addition, the vertical relations contribute as much to data reduction in this context as to simulation.

A.7 OPEN ISSUES

The concepts described in this section are but a start towards a complete methodology for evaluating C^3 systems. The three most prominent desiderata are: (a) more extensive validation that assertions 1 to 5 in subsection 3.4 are indeed true, (b) the ability to meld behavioral measures with models for structures which change over time, and (c) development of analytical tools for evaluating transient values of measures (ensemble averages) without resorting to the expense of simulation or exercise.

A.7.1 Validation

The concepts described herein have been applied to a generic air defense mission (see Vol. II). While no evidence was produced contradicting any of

the assertions presented above, one exercise of a method does not constitute proof of the generality of the method. Air defense is a well studied, highly structured mission area. Whether or not (apparently) less structured missions such as cover and deception can be reduced to STAPN models remains to be seen.

In addition, one drawback of the concepts constucted above was clearly demonstrated in both the air defense analysis and in another, similar study of NORAD's Missile Warning Center: complexity dominates any manual effort. While the hierarchical structure of a model helps manage complexity, there is a limit to the number of pages of diagrams which any person can manipulate simultaneously. Purely manual validations of these concepts are likely to be expensive and frustrating. Fortunately, the rigorous framework common to the concepts opens opportunities for partial automation of STAPN modeling and measure generation.

A.7.2 Structural Dynamics

Structural measures convey how the interconnections of a system (i.e., the topology of a STAPN model) change over time due to failures, repairs, enemy action, or reconstitution. Current evaluation technology generally limits us to evaluation of behavioral measures in the context of one specific structure. If the structure of a system changes, then we typically re-evaluate the behavioral measures, once for each system structure. In cases where behavioral measures are in fact evaluated along with structural changes, little work has been done to carefully define the measures -- time averages may not make sense because of intervening structural changes, and ensemble averages may be questionable because the system structure may change at different times in different replications.

Unfortunately, the changes in system structure are often driven by the behavioral events. If a C^3 system permits attackers to penetrate defenses frequently, then the time between structural changes caused by enemy actions will be short. Integration of (transient) behavioral measures with structural measures would be much more realistic than present techniques.

A.7.3 Continuous Time Models

Finally, the entire process of evaluating C^3 systems should be made more cost effective, so that taxpayers' dollars are spent more wisely and so that more comprehensive evaluations can take place, resulting in more effective fielded systems. Since time varying, ensemble averages provide the most insight into system behavior, this means that either simulation technology should be improved, or that analyical techniques which directly compute ensemble averages should be developed.

The latter notion may not be unrealistic in the context of STAPN models. Recall that the horizontal and vertical relations can be written for ensemble averages as well as for steady state measures. As with steady state queuing analyses, there are not enough horizontal relations to completely detemine

values for the measures. However, it may be possible to augment the canonical horizontal relations with other relations derived from timing models or decision rules. Ideally, these additional relations would take the form of delay-differential equations, so that they are compatible with the structure of the canonical relations. With certain approximations, it may be possible to reduce the completed set of delay-differential equations to a set of ordinary differential equations, for which numerous solution techniques exist.

APPENDIX B

ILLUSTRATIONS OF PETRI NET MODELS REPRESENTING HUMAN-MACHINE INTERACTION

B.1 PURE* PETRI NETS

Pure Petri nets have nodes (places, transitions) and arcs that connect the nodes to form a network. Often the place nodes represent pre-conditions or post-conditions of some event, and the transition nodes represent the event. In other models, the places represent processes, and the transitions mark beginning-of-processing and end-of-processing. Tokens in places in the network indicate that the pre-conditions (or post-conditions) are true, or that the processing is in progress. Transitions consume tokens from their input places and create tokens for their output places; this can indicate for example, that pre-conditions no longer hold, or that processing has begun. Places have only one output transition.

Pure Petri nets are used to model systems for which it is necessary to represent the coordination of resources or processes. For example, consider simple message-handling by an operator as shown in Fig. B-1.

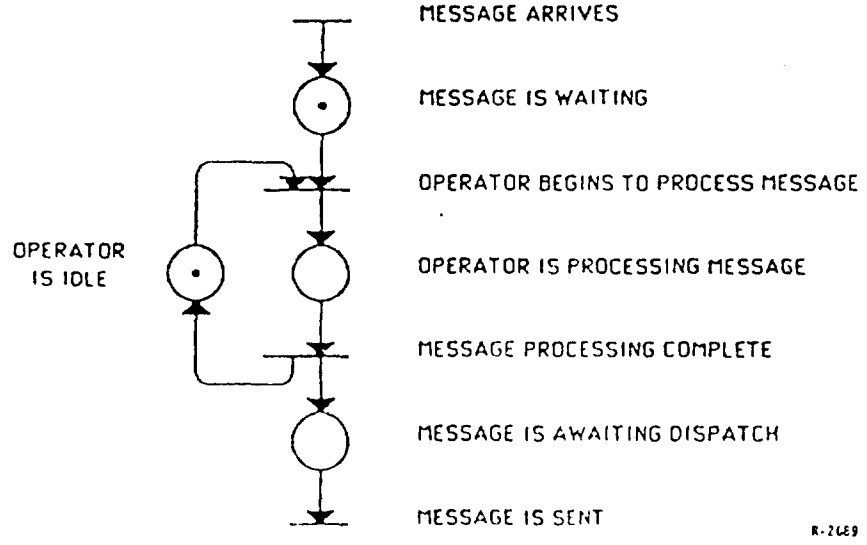


Figure B-1. Simple Message-Handling by an Operator

*i.e., without extensions. "Pure Petri nets" refers to C.A. Petri's original description (1962), as opposed to "extended Petri nets," which include modifications to increase the expressive power of the methodology.

In the above example, message-processing can begin because a message is waiting and the operator is idle.

In Fig. B-2, a message is waiting -- it cannot begin processing because the operator is not idle.

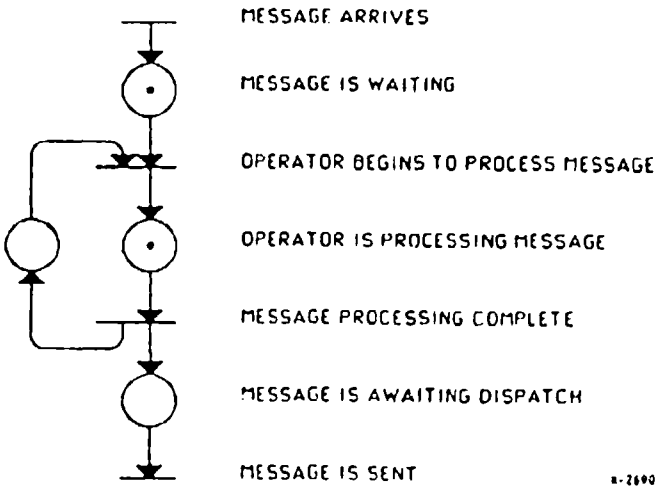


Figure B-2. Message Waiting

In Fig. B-3, two operators are available. Also, the place where messages wait (prior to sending) has room for only three messages.

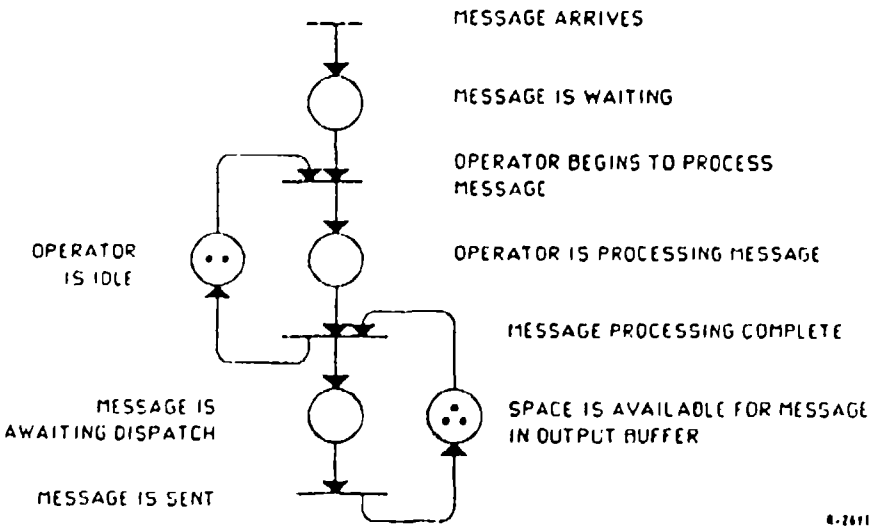


Figure B-3. Two Operators, Output Buffer of Capacity 3

Figure B-4 depicts message handling with confirmation required.

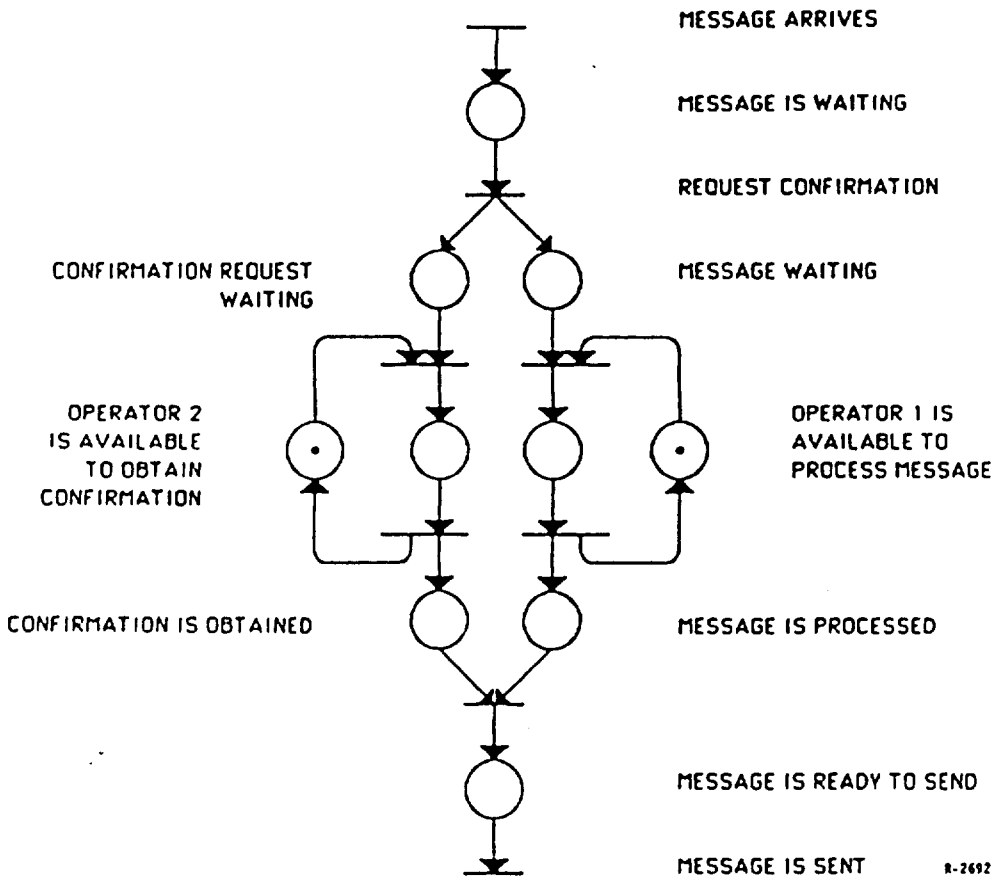


Figure B-4. Message Processing With Confirmation

In the above figure, the message is ready to send when the message is processed and confirmation is obtained.

B.2 LIMITATIONS OF PURE PETRI NETS

B.2.1 No Variation of a Token's Path

It is not possible to model a branch in a token's path (because a place may have only one output transition).

Consider our first example (Fig. B-1): simple message handling. We know that messages are sometimes garbled. Suppose (for simplicity) that one message in three is garbled and needs to be queued up for reprocessing. How might we model this? Consider Fig. B-5.

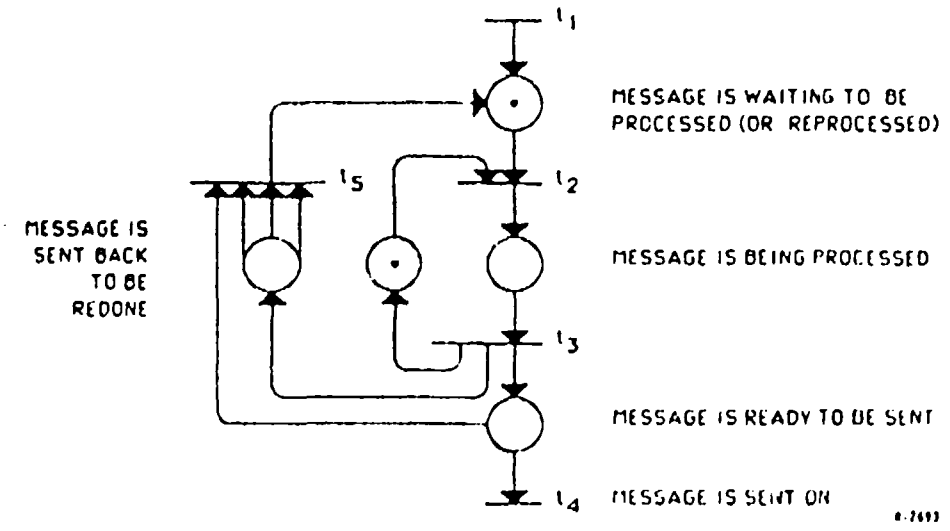


Figure B-5. Message Reprocessing

A message that is ready to be sent may follow one of two paths: it may be sent on, or sent back to be redone. In this simple model, every third message is sent back -- will it work? Only if it is somehow guaranteed that whenever t_4 and t_5 are both enabled, t_5 will fire first. t_4 and t_5 are in conflict, since firing t_5 disenables t_4. The problem of transitions in conflict will be considered later.

B.2.2 No Explicit Consideration of Timing Effects

In pure Petri net modeling, processes are considered to take some indeterminate amount of time. The issue is not one of using resources efficiently, but preventing situations such as bottlenecks or deadlock. In parallel processing situations, tokens may pile up, or resources (operators) be idle, if the parallel processes take different amounts of time.

Consider the example in Fig. B-4: message processing with confirmation. Suppose it takes 60 seconds to process a message, but two minutes to obtain confirmation. The model given still represents the necessary coordination correctly - a message will not be sent on until it has been processed and confirmed. But if messages arrive at a rate faster than one every two minutes, the processed messages will pile up awaiting confirmation. One can easily prevent this as shown in Fig. B-6.

In Fig. B-6, the place p_1 on the right prevents either task from beginning until a prior message has been processed and confirmed. However, now one operator will be idle 50 percent of the time, and messages may pile up awaiting processing. The model may not give us any indication. (The place p_1 is merely redundant.)

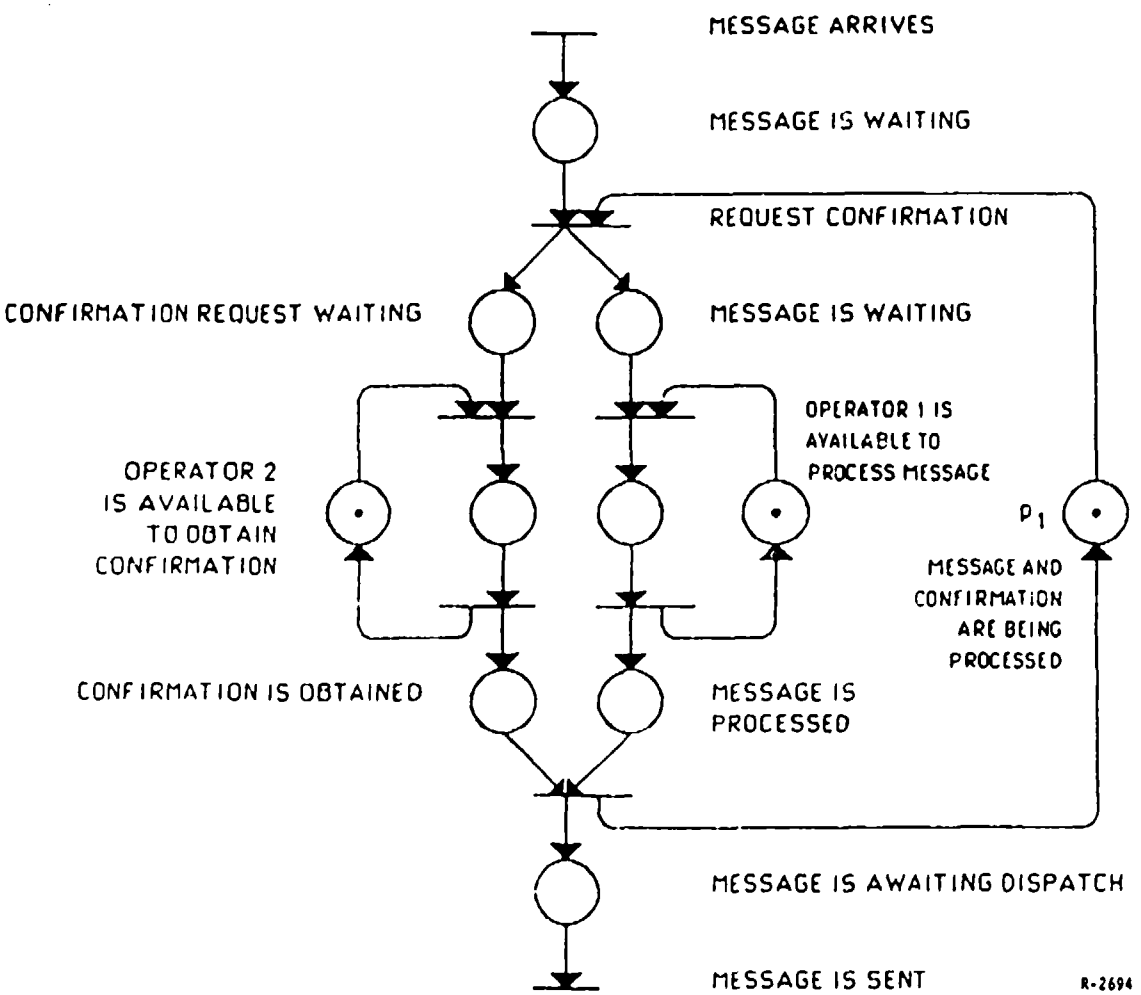


Figure B-6. Message Processing With Confirmation

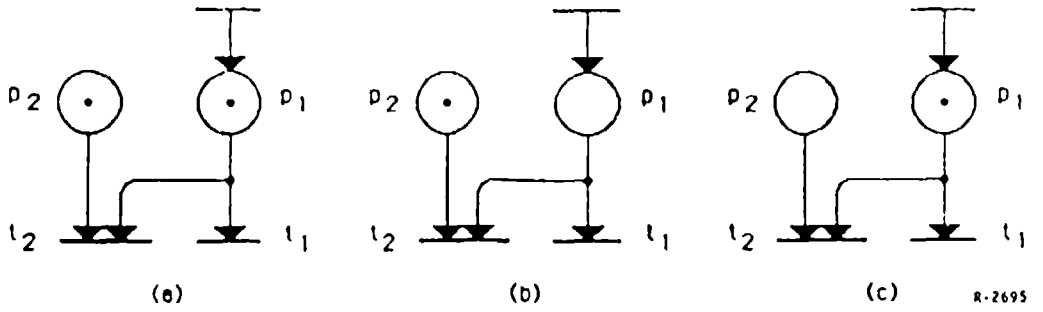
B.3 EXTENSIONS OF PETRI NETS

B.3.1 Explicit Consideration of Branching* in the Net

To model systems in which a token may travel from a place to one of two or more transitions, we allow explicit branches in the links between places and transitions (Fig. B-7). Transitions may then "share" common input places: two transitions might be enabled in such a way that firing one disenables the other.

Transitions shown in Fig. B-7(a) are in contention; we need a decision rule associated with p_1 to decide where the token should go in the event that both transitions are enabled: e.g., "Every third token to t_2" or "Go to t_2 if t_2 is enabled."

*Petri nets without branches are called Decision-Free.



In (a) the decision rule associated with p_1 determines which transition will fire.

In (b) and in (c) there is no contention, so the decision rule is not used.

Figure B-7. Transitions in Contention

Consider the simple message handling example (Fig. B-1), but suppose there are two classes of messages: status messages and alarm messages; naturally, the operator should attend to alarm messages as they arrive. This is modeled as in Fig. B-8.

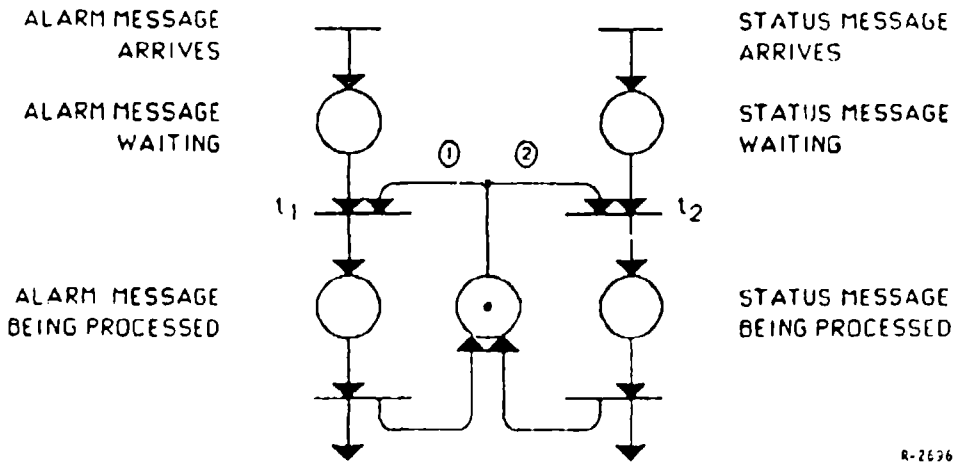


Figure B-8. Message Handling With Priority

The decision rule is: token goes to t_1 if both t_1 and t_2 are enabled.

Consider the case shown in Fig. B-9 when one message in three is garbled:

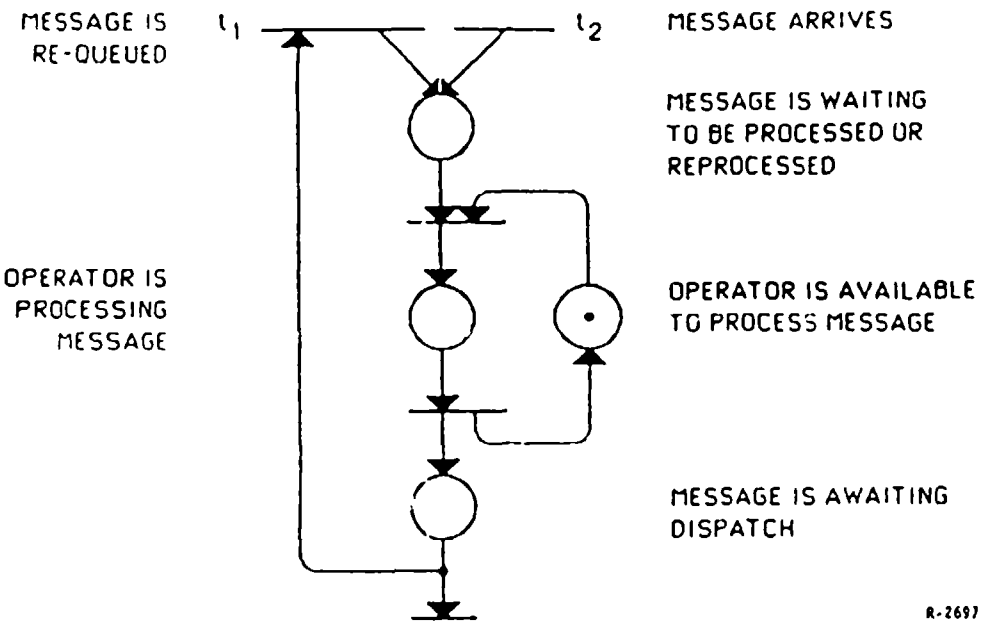


Figure B-9. Message Handling With Reprocessing

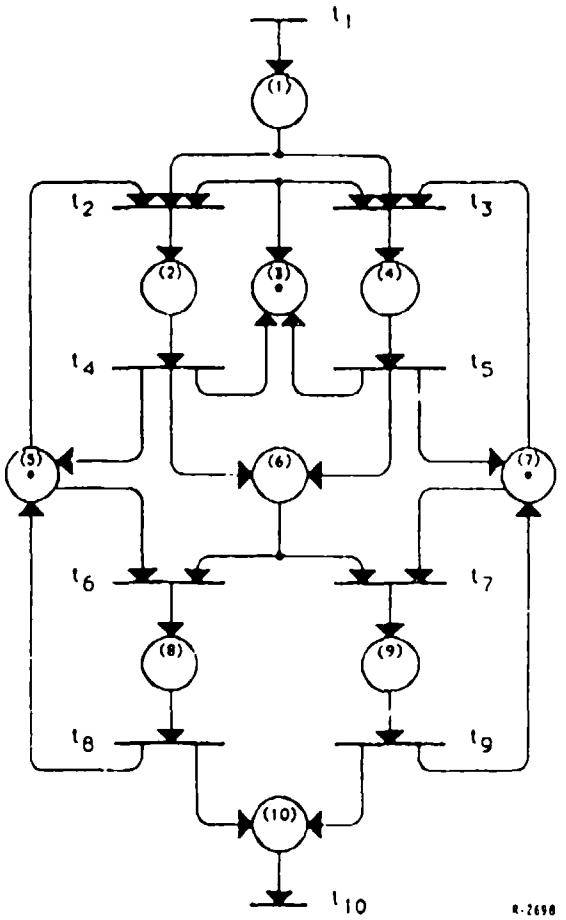
The decision rule is: every third token goes to t_1 (in a random way).

B.3.2 Explicit Consideration of Timing in the Net

Consider the message processing center example in Fig. B-10. Two operators (I & II) process two message types (A & B). All messages must be confirmed by telephone (1 line) before processing. Either operator may confirm a message, but type As are processed by operator I and Bs by II. Thirty percent of messages are type A.

Each transition assigns a time (possibly zero) to tokens that it creates: the token is unavailable* for that time in the subsequent place, possibly disenabling subsequent transitions (see Fig. B-11).

*An unavailable token is indicated by "0."



t_1: message arrives
t_2: Operator I begins to confirm message
t_3: Operator II begins to confirm message
t_4: Operator I completes confirmation of message
t_5: Operator II completes confirmation of message
t_6: Operator I begins to process message type A
t_7: Operator II begins to process message type B
t_8: Operator I completes processing of message type A
t_9: Operator II completes processing of message type B
t_{10}: message is sent

p_1: message is awaiting confirmation [Rule: output to whichever transition is enabled; t_2 if both are enabled]
p_2: message is being confirmed by Operator I
p_3: telephone is available [Rule: same as 1]
p_4: message is being confirmed by Operator II
p_5: Operator I is idle
p_6:* message is waiting processing [Rule: send 30 percent of tokens to t_6, in a random way]
p_7: Operator II is idle
p_8: message type A being processed by Operator I
p_9: message type B being processed by Operator II
p_{10}: message is waiting to be sent.

Figure B-10. Message Processing Center*

*On any given day it is likely that the mix of messages (As and Bs) will only approximate 30 percent; the actual distribution will vary. The decision rule at (6) will reflect this randomness.

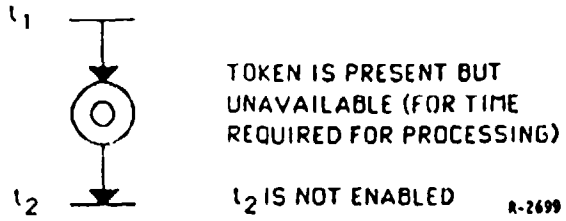


Figure B-11. Unavailable Token

e.g., Consider the message processing with confirmation example (Fig. B-6): processing takes 60 seconds but confirmation takes two minutes (see Fig. B-12):

B.3.3 Considerations of Stochastic Timing

In real life, tasks rarely take precisely 60 seconds, or exactly two minutes. Usually tasks require some time like "two minutes, give or take ten seconds," each repetition requiring a slightly different amount of time. Stochastic-timed Petri nets are as discussed above except that the times assigned are chosen from an appropriate probablity distribution, so that the times given in the example would be "about 60 seconds," "near two minutes" and so forth, depending on the exact times assigned.

B.4 FROM PETRI NETS TO QUEUING REPRESENTATIONS: STOCHASTIC, TIMED, ATTRIBUTED PETRI NETS (STAPNs)

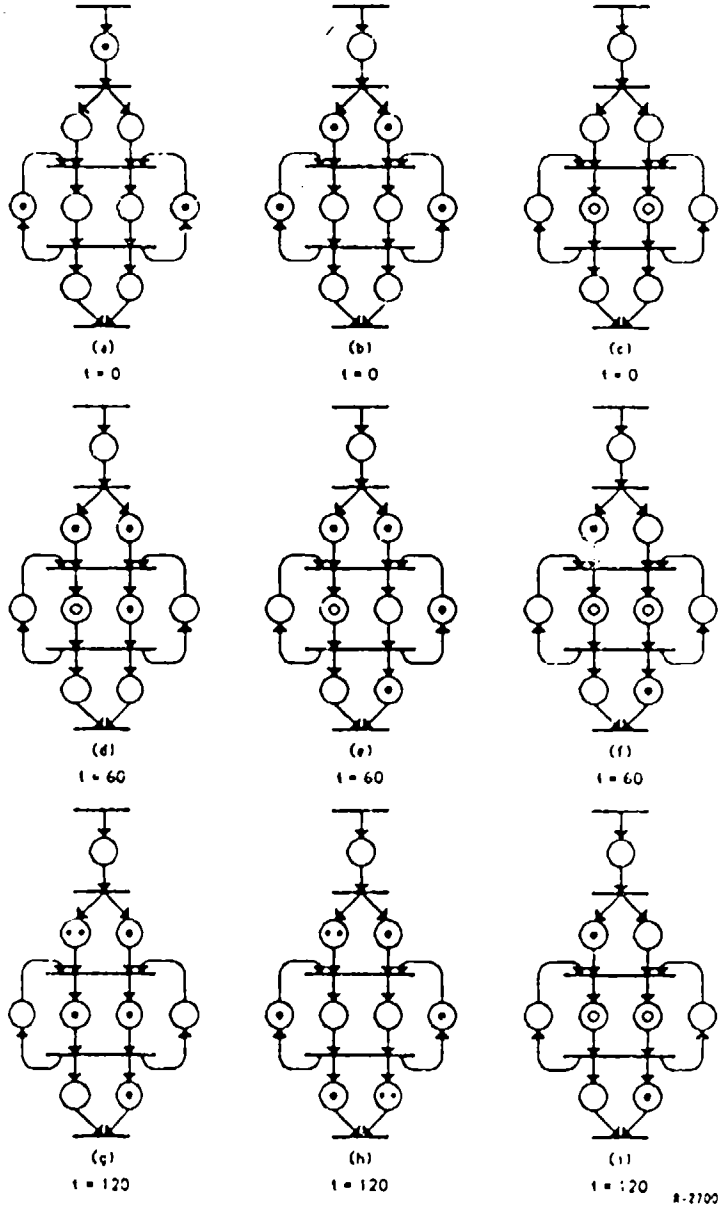
Section 4 of the text discusses queuing theory approaches to IAT. The following examples illustrate STAPN representations of simple queuing systems.

B.4.1 Simple G/G/1 Queuing Model

The notation X/Y/N is used to describe a queuing system, where X indicates the nature of the arrival process, Y indicates the nature of the service time distribution, and N the number of servers.

A G/G/1 queuing system has a general (that is, any) arrival process, a general service time distribution and one server: a G/G/1 queuing system is any single server queuing system.

Figure B-13 shows a block diagram of a single-server queue. Figure B-14 shows a Petri net representation of arrivals into the queue, and Fig. B-15 shows a Petri net representation of the queue and service facility. Figure B-16 shows the Petri net representation of the complete system.



(a) message arrives
(b) message and confirmation request are routed
(c) message processing begins and confirmation is being obtained
(d) message processing concludes, confirmation still being obtained (in the meantime another message arrives)
(e) message waiting to be sent
(f) processing next message begins
(g) confirmation is obtained ... next message processing also finished ... another message has come in
(h) begin processing third message but only second confirmation
(i) first message gone ... already we see messages piling up in one place while confirmations are accumulating further back in the system: the model shows us that this is happening.

Figure B-12. A Timed Petri Net

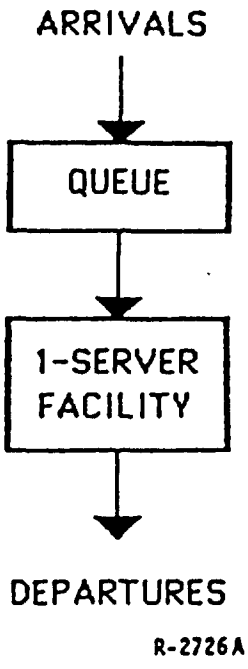


Figure B-13. Block Diagram of a Single Server Queue

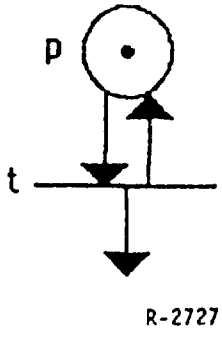
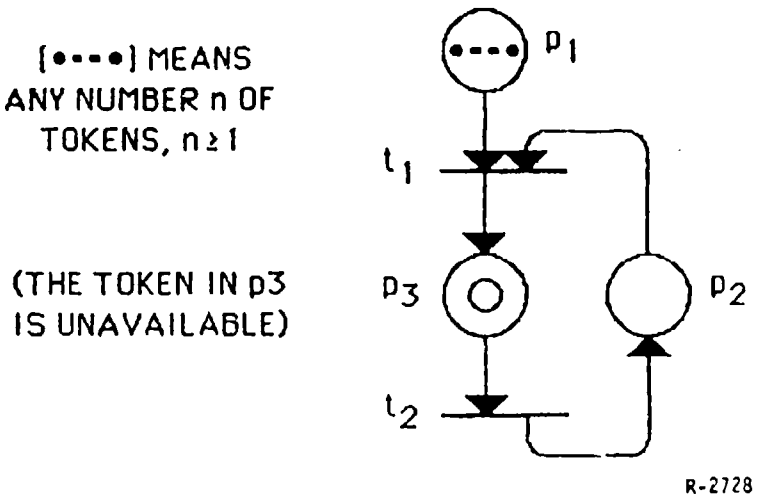


Figure B-14. Generating Arrivals into the Queuing System

The transition fires whenever the token in the place is available. It consumes this token and creates two output tokens. The token that is output to the place p is unavailable for a time that is determined by the inter-arrival distribution. [For example, if customers arrive "every five minutes, give or take a minute or so," then the time this token is unavailable should be drawn from a uniform distribution with mean 5 and standard deviation 1.] The other token is available immediately.



a token in p_1 indicates "a customer is waiting in the queue"

a token in p_2 indicates "the server is available"

a token in p_3 indicates "the server is attending to a customer"

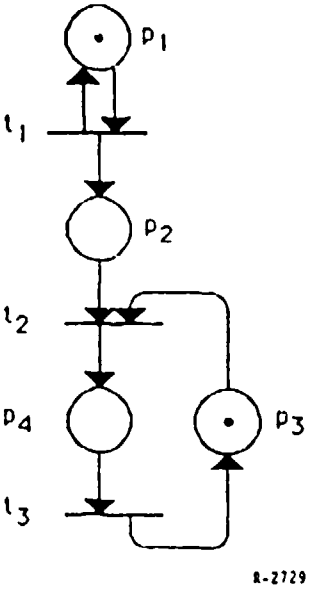
the firing of t_1 indicates "service starts for one customer"

the firing of t_2 indicates "service finishes for one customer"

t_1 fires when tokens are available in p_1 and p_2. It consumes these tokens, so there is one less token (maybe none) in p_1, and no token in p_2. [This corresponds to one less customer in the queue, and the server no longer available.] It creates one token which is unavailable for a time determined by the service time distribution, and this token goes into place p_3, indicating service is in process. [For example, if service times average "three minutes give or take half-a-minute" then the time this token is unavailable should be drawn from a distribution with mean 3 and standard deviation .5.]

t_2 fires when the token in p_3 becomes available, indicating service is ended. It consumes the token in p_3 and creates a token for p_2 which is immediately available.

Figure B-15. Queuing and Service



t_1: arrival
t_2: begin service
t_3: complete service

p_1: create arrivals
p_2: queue
p_3: server available
p_4: service in progress

Figure B-16. Petri Net Representation of Single Server Queuing System at Time T=0.

B.4.2 Simple G/G/N Queuing Model

"G/G/N" refers to any multi-server queuing model. This model is the same as the preceding except that p_3 contains N tokens (shown as "N•" in Fig. B-17):

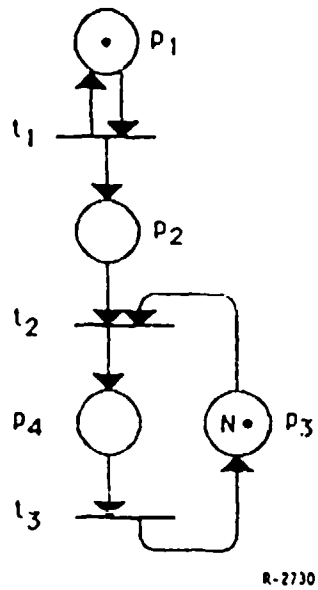


Figure B-17. Petri Net Representation of Multiserver Queuing System at Time T=0

In the following (Fig. B-18), a 5-server facility is serving three customers and no customers are waiting:

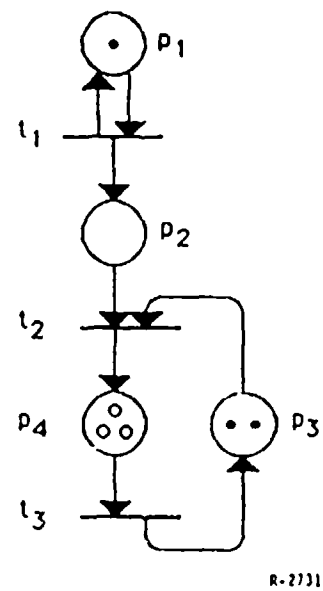


Figure B-18. Five-Server Queuing Model

B.4.3 Simple G/G/N Queuing with Type I Reneging

A queuing system exhibits reneging if a customer leaves the system before service has been completed. In many situations, a customer will not wait indefinitely, but will always complete service once service has begun. This customer leaves the system from the queue only; this is Type I reneging (Fig. B-19).

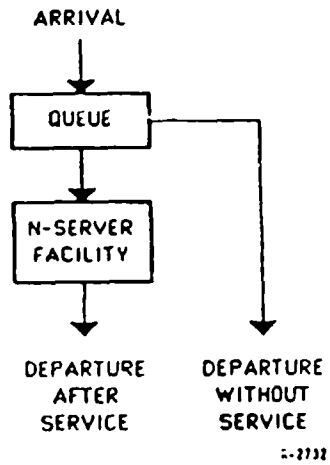
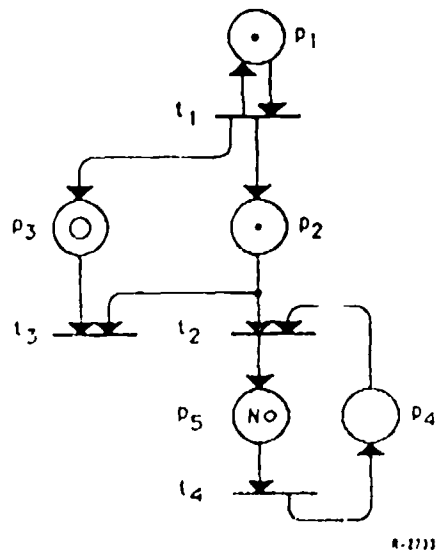


Figure B-19. Block Diagram of Multiserver Queuing System with Type I Reneging

Using Petri net notation, we can represent the G/G/N case by adding a place, a transition and a branch (Fig. B-20):



t_1: arrival
t_2: begin service
t_3: reneg
t_4: end service

p_1: generate arrivals
p_2: queue available for service
p_3: queue waiting to reneg
p_4: server available
p_5: service in progress

Figure B-20. Petri Net Representation of Figure F-19

In the model shown in Fig. B-20, t_1 creates two output tokens: the token that is forwarded to p_2 is available immediately; the token that is forwarded to p_3 is unavailable for the reneging time of this system. [If every customer will wait exactly 5 minutes, then this token is unavailable for 5 minutes.] As shown, no servers are available (no tokens are available in p_4).

Place p_2 has two output transitions, t_2 and t_3. The transition that is enabled first will consume the token. Thus if a server becomes available before the reneging time is elapsed, then t_2 will be enabled and a token will be passed to p_5, indicating service in progress. If the reneging time elapses first -- (1) the token in p_3 will become available; (2) t_3 will be enabled; and (3) the tokens in p_2 and p_3 will be consumed. This situation represents the fact that the customer is never serviced.

Note that there is a problem with this model as it stands. Every time a token is consumed by t_2 (service begins), the token in p_3 that represents a reneging time will become available at a later time. Subsequent tokens in p_2 will then "reneg" immediately.

The problem can be corrected by adding another place, another transition, and another branch (Fig. B-21):

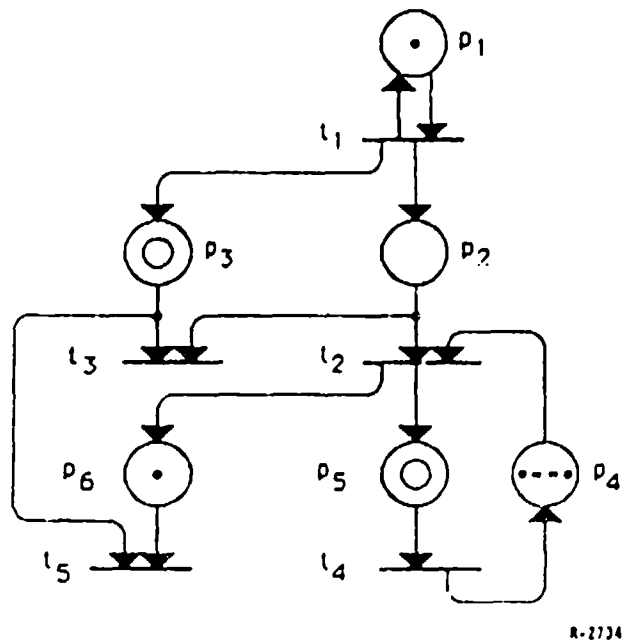


Figure B-21. Final Version of Figure B-20

Now t_2 also creates two output tokens and the token in p_6 is available immediately. It represents the fact that processing has started for this customer. When the reneging time is elapsed, the token in p_3 becomes available, t_5 is enabled, and the tokens in p_3 and p_6 are consumed.

A problem still remains. Consider a single server system: one customer being served, one customer waiting service, and the reneging time for the first customer has elapsed (Fig. B-22):

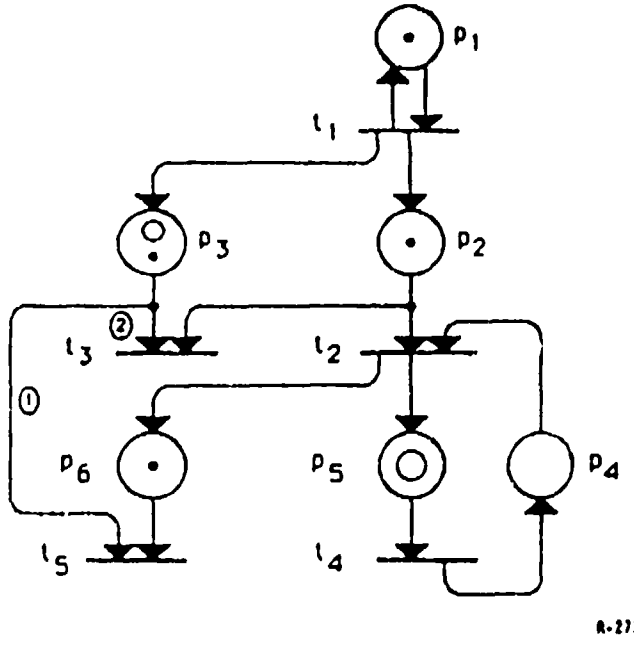


Figure B-22. Contention Between Transitions t_3 and t_5

Now both t_3 and t_5 are enabled. Firing either will disenable the other; these transitions are in conflict. We must have a decision rule associated with p_3 which directs the output of its tokens when both output transitions are enabled. Here we choose a simple deterministic rule: t_5 is always enabled if there is a conflict.

Actually, if our implementation allows events to occur "simultaneously," then we need a decision rule for p_2 as well. Suppose in the above example (after firing t_5) that the server becomes available (service time in p_5 elapsed) at precisely the same instant that the token in p_3 becomes available (reneging time elapsed). Figure B-23 illustrates this situation:

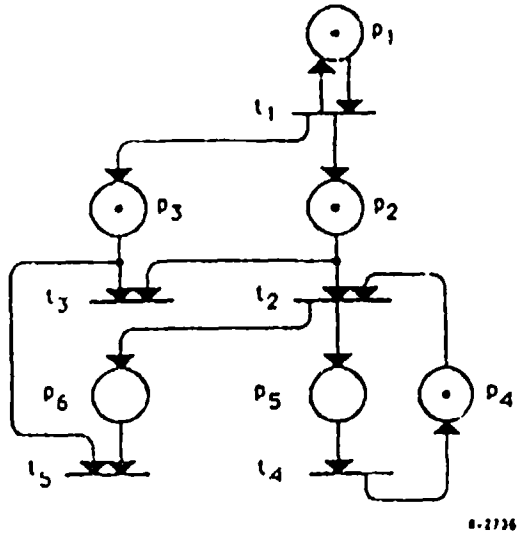


Figure B-23. Contention Between Transitions t_2 and t_3

Now t_2 and t_3 are in conflict. Here the decision rule we choose for p_2 may be deterministic: "always enter service" (t_2 is enabled) or "always reneg" (t_3 is enabled). Or, depending on the situation being modeled, we may feel that there is some probability that the customer will reneg, and thus the decision rule may have probabilistic content: "chances are 60% of customers will reneg;" that is, in a random way, enable t_3 60% of the time.

B.4.4 G/G/N Queuing with Type II Reneging

In some situations, a customer will leave the system after some length of time (the reneging time) regardless of whether service has begun or not. A system exhibits Type II reneging if customers will reneg from the queue or leave while being served. Figure B-24 shows the block diagram to represent this case and Fig. B-25 presents the associated Petri-net model.

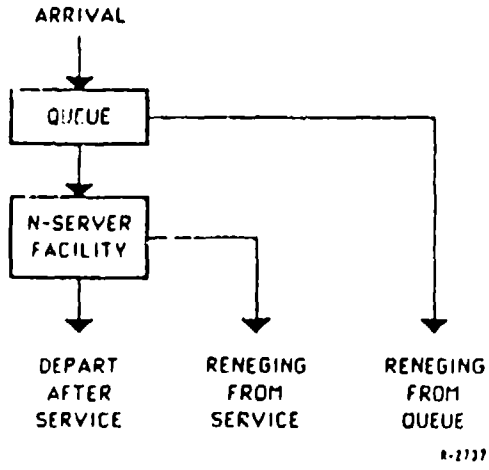
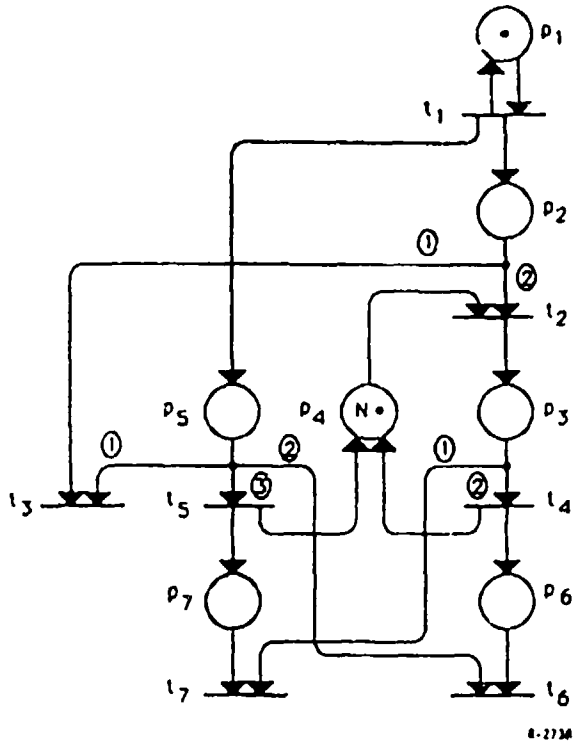


Figure B-24. Multiserver Queuing System with Type II Reneging



t_1: customer arrives:
- p_1 token available after interarrival time.
- p_2 token available immediately.
- p_5 token available after reneging time.

t_2: service begins:
- p_3 token available after service time.

t_3: customer renegs from queue.

t_4: ·vice ends:
- h tokens available immediately.

t_5: reneging time cc pletes:
both tokens available immediately.

t_6: customer departs having completed service.

t_7: customer departs having reneged from service.

p_1: token generator.

p_2: customer in queue.
Rule: go to t_3 (if both t_2 and t_3 enabled) [this could vary].

p_3: service in progress or completed.
Rule: go to t_7 if enabled.

p_4: server is available.

p_5: reneging time is elapsing or has elapsed.
Rule: go to t_3 if t_3 is enabled; else go to t_6 if t_6 is enabled; else go to t_5.

p_6: service time has completed and reneging time has not.

p_7: reneging time has completed and service time has not.

Figure B-25. Petri Net Representation of Figure B-24

B.5 EXAMPLE SHOWING USE OF DECISION RULES IN PETRI NET MODELING

A simple model of an operator managing two streams of messages is shown in Fig. B-26 below:

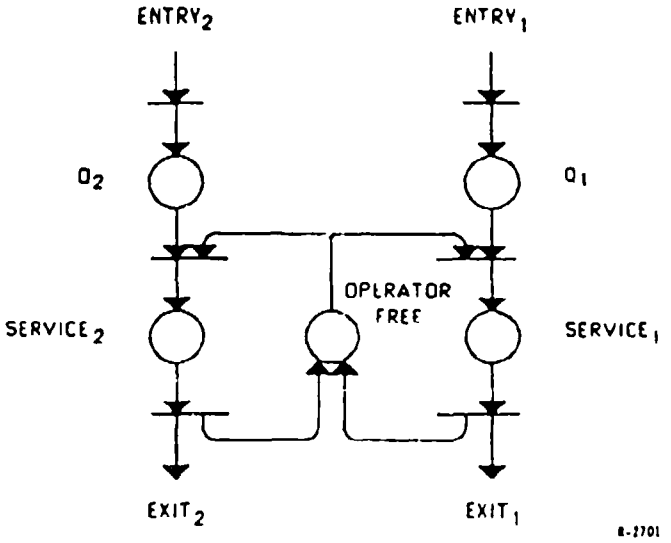


Figure B-26. Model of an Operator Servicing Two Task Streams

If the decision rule at the branch is "allocate the operator (token) to any nonempty queue of messages, this model is complete (tokens flow through the branch to any enabled transition at the exit of Q_1 and Q_2). If the decision rule is "allocate the operator (token) to the longest nonempty queue, with ties broken in favor of Q_1", then there is a tradeoff. If this network description is used, then the decision at the branch is made based on state information beyond the enablement status of the transitions terminating the branch. The number of tokens in Q_1 and Q_2 must be compared for the decision to be made and neither Q_1 nor Q_2 is directly connected to the branch. Hence the simplicity of the network gives rise to a coupling between distant places and decision rules.

The second decision rule can be implemented by a net where such distant couplings are absent, albeit at a cost of topological complexity. The decision rule can be decomposed as:

IF	$\|Q_1\| - \|Q_2\| > 0$	THEN	serve a token in Q_1	(P1)
ELSE IF	$\|Q_2\| - \|Q_1\| > 0$	THEN	serve a token in Q_2	(P2)
ELSE IF	$\|Q_1\| + \|Q_2\| > 0$	THEN	serve a token in Q_1	(P3)
ELSE	wait.			

where $|Q_1|$ is the number of tokens in Q_1 at the time an operator token becomes available in the "OPERATOR FREE" place. Adding structures to evaluate predicates P1-P3 directly in the Petri net* gives a more complex net, but decision rules are only dependent upon the enablement status of transitions connected to the corresponding branch.

$$\text{In Fig. B-27,} \quad |\text{EXCESS } Q_1 \text{ OVER } Q_2| = |Q_1| - |Q_2|$$

$$|\text{EXCESS } Q_2 \text{ OVER } Q_1| = |Q_2| - |Q_1|$$

$$|\text{TOTAL}| = |Q_1| + |Q_2|$$

Tokens become available immediately (after zero delay) in all but the SERVICE_i and OPERATOR FREE places.

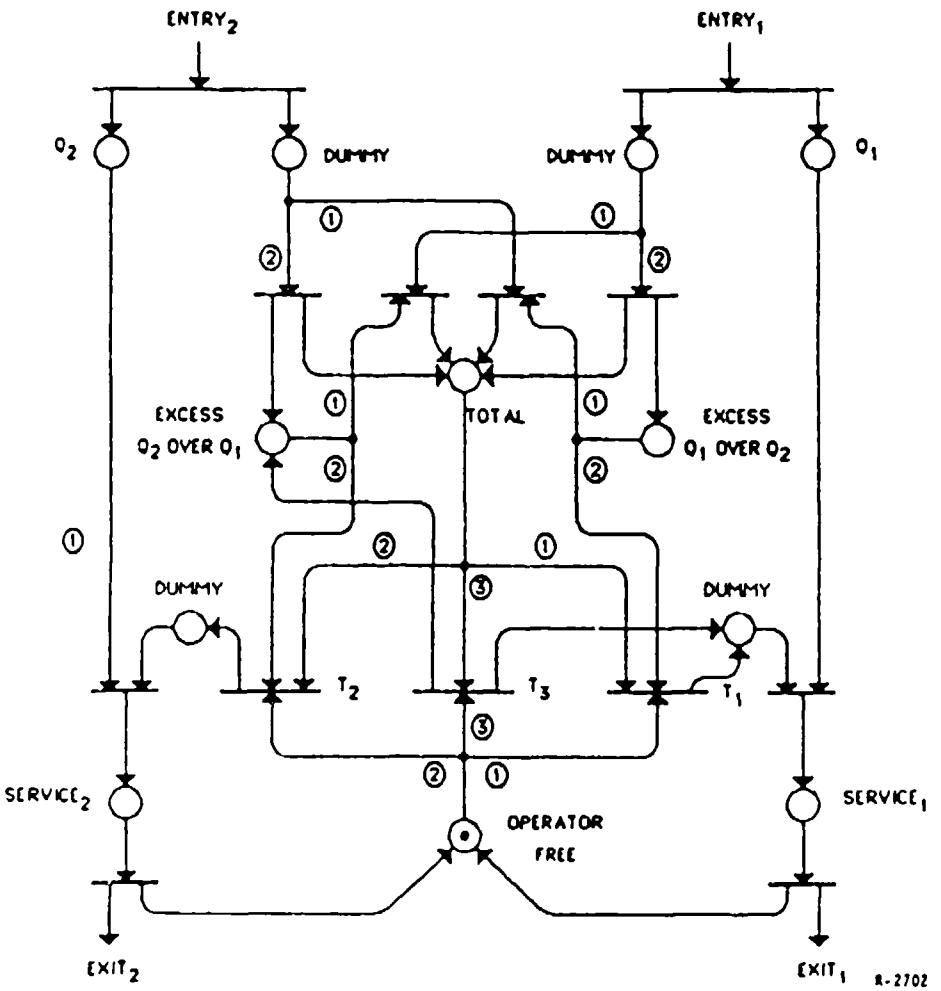


Figure B-27. Adding Structure to Evaluate Decision Rules

*See Fig. B-27.

Decision rules are all of the form "pass the token to the enabled transition with the smallest index"; i.e., the decision rule following OPERATOR FREE is:

IF	|excess Q1 over Q2|	>0	THEN	FIRE T_1
ELSE IF	|EXCESS Q2 over Q1|	>0	THEN	FIRE T_2
ELSE IF	|TOTAL|	>0	THEN	FIRE T_3
ELSE			WAIT.	

APPENDIX C

PROCEDURES AND GUIDELINES FOR APPLYING IAT TO REAL-WORLD SYSTEMS

C.1 INTRODUCTION

Using modeling notations such as frame/slot and Petri nets to describe real-world systems requires a high-level understanding of system structure, purpose, and relationships among elements within a system. The components of the IAT conceptual framework have been developed to aid analysts in gaining this understanding.

Although the focus of work has been to define and refine the modeling techniques per se, the needs of (prospective) IAT users, who would be required to learn the methods themselves and apply them to operational systems, have also been identified. The subsections below include generic guidelines, intended to support analysts who wish to learn and use IAT techniques for analyzing human/system performance of C^3 systems. Examples used in stating these procedures have been drawn from ALPHATECH experience to date on modeling the Missile Warning Center (MWC) at NORAD Cheyenne Mountain Complex (NCMC) and a generic Air Defense System (see Vol. II). The guidelines are presented here from the broadest level of applicability -- model development in general -- to a more specific approach geared to analyzing decisionmaking within operational environments such as those found at MWC.

C.2 GENERAL GUIDELINES FOR IAT MODEL DEVELOPMENT AND USE

1. Define "the Problem:" Why

   - Who's asking the question -- Perspective

   - When is the answer needed -- Scope

2. Review the Facts: What

   - Do not reinvent wheels or probe blind alleys

   - Do not assume non-available information

3. Identify the Analogies: How

   - What is this "like" (or not like)

   - Where does the analogy begin to fail

4. Develop the Model

   - Determine the degree of decomposition needed -- scope

   - Identify the variables

5. Formalize the Mathematical (Petri Net) Representation of the Model

   - Develop the complete model

   - Define a complete set of measures

6. Parameterize the Model

   - Estimate internal variables (e.g., coefficients based on resource characteristics and capabilities)

   - Estimate external variables (e.g., scenario input based on threat and environment characteristics)

7. Solve the Model; Exercise the Representation

   - Analytically: Given X, find Y using PERT/CPM or Queuing Theory Methods

   - Numerically: Generate X, compute Y

8. Interpret Results

   - Verification

   - Validation

9. Sensitivity Analysis

   - What is critical/trivial

   - How close is good enough

10. Execute Production Runs: Prediction/Estimation

    - Interpolate: Precision

    - Extrapolate: Speculation

C.3 GENERAL RULES TO HELP FOCUS DATA COLLECTION -- FOR BUILDING IAT STRUCTURAL AND PERFORMANCE MODELS

1. For each observed process (activity, task), identify the following:

   a. What constitutes successful completion?
      [i.e., look for GOALS in the IAT framework]

   b. Who normally carries out the process or executes the task?
      [identify responsible individuals (RESOURCES), who have specific roles in ORGANIZATIONS]

   c. Where is the process carried out?
      [e.g., workstations, display terminals, other locations (IAT RESOURCES)]

   d. How is the process done?
      [by what means, using what equipment or information]

   e. What triggers the process or permits it to start? to stop?
      [activation(s), termination(s)]

   f. If a process is interrupted, who determines whether to abandon or restart it? What factors influence that decision?

   g. After interruptions, how is restart accomplished?

      - with no set-up, some set-up, or complete restart?

      - does set-up include activities that would not have been done otherwise?

   h. What happens when performance is blocked?

      - required inputs are not available

      - required resources are not available

      - required authorization is not available

   i. Can this process cause other processes to be interrupted? (What happens to resources?)

2. Look for information about each observed process:

   a. How often does it occur?
      [rate or time between events]

   b. How long?
      [speed or duration]

   c. How important is it to the purpose of the overall system?
      [which activities can it pre-empt ... by which other activities can it be pre-empted]

   d. Where are errors likely to occur?

3. Identify precedence constraints on processes (serial/parallel distinctions):

   a. Do things have to happen sequentially? Why?

   b. Could things be done concurrently? If so, how could concurrent execution be achieved?

4. Describe conditional dependencies (rules and factors):

   a. When/if _____, do _____, not _____

   b. Unless/until _____, do _____

C.4 MORE SPECIFIC GUIDELINES FOR BUILDING IAT STRUCTURAL MODELS*

1. Define the system or subsystem boundary: e.g.,

   a. Input bound = message receipt at MWC

   b. Output bound = CINC's decision reported outside of NCMC

2. Define the system performance criteria: e.g.,

   a. Time that first notification report is transmitted

   b. Time that last notification report is transmitted

Those items marked with "" will be treated in more detail in subsequent sections of this appendix.

3. Develop a "back-chained logic"* for the data flow sequence: viz.,

   a. Identify content of each output or product (e.g., report).

   b. Collect information about how that output is produced, and by whom* (PROCESSES, RESOURCES, and ORGANIZATIONAL* elements in IAT).

   c. Identify each input used in producing each output (IAT RESOURCES*).

   d. Find out how this information relates (i.e., links back) to the original input bound in the system (e.g., incoming message stream at MWC).

4. Develop a "forward-chained logic"* for the data flow sequence: e.g., for the MWC,

   a. Identify all other incoming input messages: their source, content, and arrival characteristics.

   b. Identify where and how message existence will first be detected (with respect to IAT RESOURCES, ORGANIZATIONS, and PROCESSES).

   c. Identify each pathway through which message contents may flow.

   d. Trace flows to specific output products or events (even though there may be no physical output that is realized; e.g., a file that gets destroyed or a message that might be ignored).

5. Identify where in the system human involvement* is required to do the following:

   a. Decide what option or action alternative to take*, particularly in situations where a decision is on the critical path for producing a product or carrying out a task.

   b. Make a judgment,* without which a data void will exist (or a required output will not be produced).

Those items marked with "" will be treated in more detail in subsequent sections of this appendix.

C.5 MORE DETAILED PROCEDURES FOR LOWER-LEVEL ANALYSES OF SYSTEM STRUCTURE

C.5.1 Questions to Support Analysis for the IAT ORGANIZATION Dimension*

1. What is your most important job? [IAT PROCESS]

   What is the purpose [IAT GOAL] of this activity?

   What do you need to do this job [inputs] and what products or services do you provide [outputs]?

2. If time did not matter, how would you know whether you succeeded or failed [measure of performance]?

3. Does it matter when the job is done [start, stop, and/or duration]?

4. Who gets the product [where does it go next]?

5. What makes up the product [what are its component parts]?

6. How is that product built [i.e., put together, assembled, generated or produced]?

7. Where do the inputs come from [ORGANIZATION, other PROCESSES, and/or other RESOURCES)?

C.5.2 Rules for Hierarchical Decomposition for the IAT ORGANIZATION Dimension

[Examples based on Missile Warning Center (MWC) and Command Post (CP) at NORAD Cheyenne Mountain Complex (NCMC).]

1. Identify first the single point of authority for the unit being decomposed.

   - for CP: Command Director

   - for MWC: Missile Warning Officer (MWO)

*Questions at this level of generality should be directed to individuals who play specific roles in organizations. The question sequence should be repeated as needed to obtain complete job inventories. Each implied reference to associated PROCESSES, RESOURCES, and ORGANIZATIONS should be pursued until elemental tasks can be identified. (An elemental task is a single action or decision by a human agent, for which a unique output can be identified.)

2. Identify subordinate linkages - to whom that individual reports and who reports to that individual.

   - for CINC NORAD: reports to NMCC; Command Director (CD) reports to CINC NORAD

   - for MWO: reports to CD; the Events Verification Officer (EVO) and Missile Warning Supervisor (MWS) report to the MWO.

3. Identify the cooperating and coordinated units.

   - Cooperating units supply inputs

   - Coordinated units are supplied outputs

4. For each identified organizational element, identify the constituent components: the positions that collectively compose this unit.

   - for CP, lists of Daily Duty Officers and Battle Staff

   - for MWC, list of crew members

5. Prepare an associated Duties List for each individual identified; this serves as a starting point (and later as summary sheet) for the function assignment matrix.

6. If an individual can be at more than one location from time to time, it may become necessary to list these sites. This would be the beginnings of a locatability matrix (especially important for recall in the transition from peacetime to wartime operations).

7. Identify whatever formal documents exist that authorize, govern, or constrain the activities of this organizational unit.

C.5.3 Rules for MAT RESOURCE Decomposition [based on MWC operations]

1. [See ORGANIZATION decomposition, derived from procedures in subsection C.5.2, for relevant crew and workstation information.]

2. For each crew position, identify all dedicated console/workstation equipment; all portable/shared equipment; and all personnel equipment items.

3. For each equipment item (portable or dedicated), identify all input/output interfaces and the characteristics of each.

4. Within each display identify:

   a. what information indicates what is being displayed and how the operator discerns that,

   b. what options can be selected and how these are made known to the operator, and

   c. the constant and variable contents of each display format (for all possible alternate formats).

5. For each control panel, identify the major function being supported and obtain the nomenclature associated with each item (and groups of items) on every panel. Indicate the actuation mode* (push, twist, flip, pull, etc.) for each item.

6. Obtain or document relative position of each item on each panel and each panel on each console using a well-defined coordinate system.

7. Identify or define the operator's work position(s) relative to the console (standing, sitting, etc.).

C.5.4 Important Questions to be Answered About Displays and Controls

1. Display Characteristics of Interest:

   a. Are contents always available? Y/N

      (If no, name the access procedure)

   b. Are changes automatic? Y/N

      (If no, name update procedure)

   c. Are detections of change highlighted? Y/N

      (If yes, by auditory alarm? Y/N)
      (If alarm no, describe visual display dynamics).

2. Control Characterestics of Interest:

   a. Are entries self-paced? Y/N

      (If no, what controls the forced pace?)

*The list should describe what action is required to activate the control; compound actions (grasp, pull, and lift) may sometimes be required; these descriptions influence time estimates.

b. Are entries fed back for verification? Y/N

c. Are entries checked for validity? Y/N

d. Are transmitted entries acknowledged?

e. Are unacknowledged messages timed out so the operator is alerted? Y/N
(If yes, how is the alert presented?)

C.6 GENERAL GUIDELINES FOR CARRYING OUT "BACK-" AND "FORWARD-CHAINED" LOGICAL ANALYSES

C.6.1 Specifying Assumptions:

1. Assume all equipment works and crew required is on-hand and performing error-free.

2. Anytime there are multiple options:

   a. list the entire set

   b. determine which dominates (by importance, frequency, or time demanded -- in that order)

   c. pursue only the dominant flow path first; pursue others later.

3. Focus on major end products being generated at intermediate stages of evolution toward output; do not explore how these are generated (only what, where, when; not how or why).

C.6.2 Guidelines for "Back-Chained" Logical Analysis

PREREQUISITES:

1. Determine the boundary of the facility/organization unit:

   Define the I/O interface (what comes in and what goes out).

2. Isolate the I/O components of interest (e.g., messages).

3. Identify where these manifest themselves inside the unit, i.e., boundaries where inputs enter and where outputs are dispatched.

IMPORTANT QUESTIONS:

1. What are the output product(s)?

   a. Name the message(s) leaving the facility.

   b. Enumerate variable contents of each.

   c. Identify the basis for determining which message leaves in cases where more than one can (e.g., format selection).

   d. For any message, determine what controls its departure.

2. How is each output product composed? [i.e., what are the constituents of each output?]

   a. For each variable from 1b above, identify the exhaustive set of mutually exclusive alternatives that constitute content opinions.

   b. Determine who makes the selection; then ask how that selection is done.

   c. For each information item or condition that serves as an input, identify other items that can be associated as outputs (within or outside the system boundary).

3. For each identified input, repeat the process described in steps 1 and 2.

   Quit when all inputs can be viewed as output products from facilities which lie outside the boundaries of interest.

C.6.3 Questions for "Forward-Chained" Logical Analysis

1. Identify all input messages, their source and initial destination.

2. Determine buffer characteristics: how many messages can be held and for how long; when/how might data be lost?

3. What indicates new/more data are available so that processing may continue?

4. What are the processing stages between receiving, storing, and destroying data or information?

5. How might the information get used (for what, in what, by whom, and accessed how)?

6. When/how do old files get purged (by whom, on whose authorization)?

7. Can information losses be recovered and if so, how; if not, what is the impact?

C.7 PROCEDURES FOR ANALYZING HUMAN INVOLVEMENT: SITUATIONS REQUIRING HUMAN DECISIONMAKING AND JUDGMENT (From Section C.4, No. 5)

C.7.1 Decisionmaking

For each decision that is identified:

1. Who normally has decision authority (and under what conditions)?

2. What are the action alternatives from which selection is made?

3. What consequences are of concern?

4. What risks are perceived?

5. What information can change perceptions about the foregoing?

C.7.2 Judgment

For each judgment:

1. Who normally makes the judgment?

2. What changes when the judgment is made?

3. What information does the judge access before implementing that change?

4. What would data voids* do to the judgment process?

   a. Increase error

   b. Delay change implementation

   c. Reduce confidence

   d. Other

   e. Combinations of the above

APPENDIX D

PROCEDURES AND GUIDELINES FOR USING DATA FLOW DIAGRAMS TO DEVELOP IAT DATA

D.1 INTRODUCTION

Of the methods reviewed earlier in the IAT development effort (Gruesbeck et al., 1984),Structured Analysis (SA) appears to have high utility. SA is a manual, graphical, and narrative requirements analysis and design method. It was developed during the early 1970's as part of L.L. Constantine's Structured Design (SD) methodology, which emphasizes deriving processing requirements based on a description of total system data flows and state at any specific time (DeMarco, 1979; Rowell, 1981).

SA is based on top-down decomposition with simple graphical tools. Its use is intended to improve user/analyst communication and provide accurate and easily comprehended, structured requirements specifications as input to the design stages of the system development life cycle.

A complete SA model consists of the following components:

1. Data Flow Diagrams (DFDs)

2. Data Dictionary

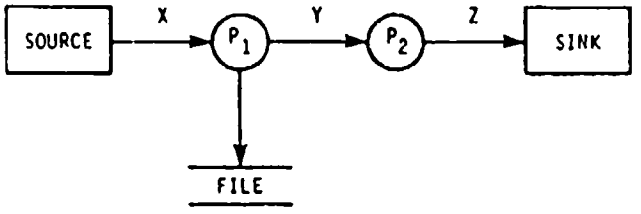
3. Mini-Specs

DFDs, which were used in the SIMCOPE and MWC applications, are network representations of a system portraying the system's component processes and their interfaces (data flows). DFDs serve to partition the system and may be used to represent manual as well as automated processes. A Data Dictionary defines each of the DFD's interfaces in terms of lower-level data flows and/or more primitive data elements. Mini-Specs describe the lowest-level processes in the DFDs using tools such as Structured English, Decision Tables, and Decision Trees (Rowell, 1981).

> Note that for IAT applications, based in part on the SIMCOPE and MWC validation studies, we are advocating the use of DFDs along with frame/slot and Petri net notations. Frame/slot and Petri net modeling techniques provide more flexible and comprehensive formats for decomposing system constituents into lower-level processes than do the (often elaborate) narratives required for Data Dictionary and Mini-Specs.

D.2 PROCEDURES FOR DERIVING DATA FLOWS USING DEMARCO DFDs

The basic components of DFDs and their application to manned C^3 systems have been presented elsewhere (Kornfeld, 1984; Kornfeld et al., 1985). The focus of the discussion here is rather to summarize guidelines for helping analysts use DFDs along with other IAT modeling tools. For this purpose, only a brief listing of DFD components appears below (Fig. D-1), with more emphasis on notational conventions.



COMPONENTS

1. Data flows, represented by labeled arrows; (X,Y,Z)
2. Processes, represented by circles; (P_1,P_2)
3. Files or Data Stores, represented by straight lines; FILE
4. Data Sources or Sinks, represented by boxes.

R-2311

Figure D-1. DeMarco Data Flow Diagram

D.2.1 Data Flow Notational Conventions

Data flows are represented by named arrows or vectors. They act as pipelines through which packets of data and information of a known composition can flow.* Conventions for deriving data flows include the following (DeMarco, 1979; Rowell, 1981):

1. Name all data flows - when no logical relevant name can be found for a data flow, chances are there is some error in the DFD.

*Note that DFDs do not show the system's flow of control. A data flow that is actually a signal to do something (e.g., a flow named Start-Next-Item) should not be portrayed on the DFDs. Control flows are procedural in nature and should be specified elsewhere (e.g., in Petri nets).

2. Choose a name which conveys to the DFD reader what the data is and what is known about it (e.g., Raw-Intelligence-Data becomes SAC-Interest-Data in Fig. D-2, which illustrates data analysis processes in HQSAC).

3. Insure that all data flows have different names to avoid confusion for the DFD reader.

4. Hyphenate multiple word data flows to show that they represent a single concept.

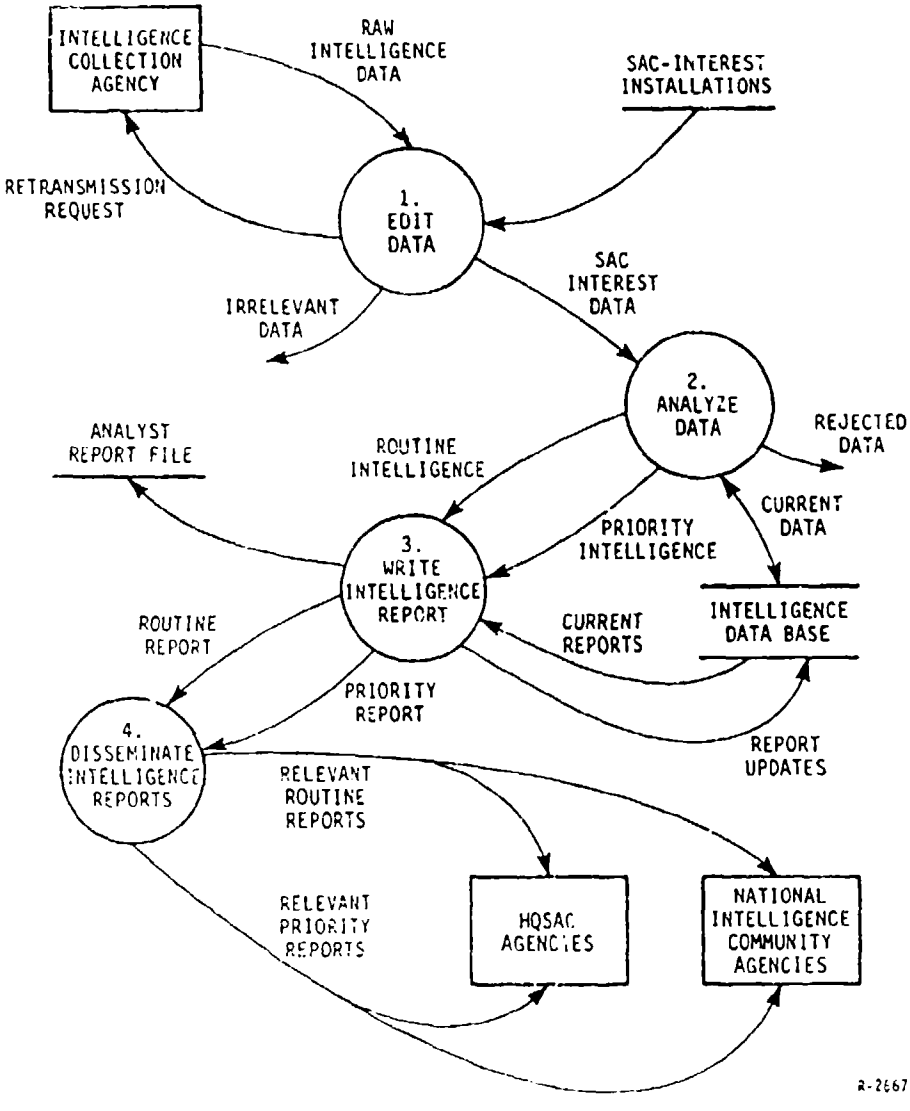


Figure D-2. Data Flow Diagram of Hypothetical Military Intelligence System (Rowell, 1981; p. 60)

5. When applicable, show data flows diverging to different elements or converging from different elements with forking or joining vectors, e.g., Relevant-Routine-Reports in Fig. D-2.

6. Let data flows into and out of simple files be identified by the file name instead of naming the data flow itself (e.g., the data flow from the SAC-Interest-Installation file to the Edit Data process in Fig. D-2).

D.2.2 Conventions for Other DFD Components: Processes, Files, and Data Sources (Rowell, 1981)

Processes are transformations of one or more incoming data flows into one or more outgoing data flows (e.g., Edit Data, Analyze Data in Fig. D-2). Processes are represented by circles which are labelled with a number and a descriptive name. The numbering system and its rationale are part of the leveled DFD concept which is explained below.

Files are temporary repositories of data which include a computer tape or disk, an index file in someone's desk, a computer database, or even a waste-basket. They are represented by a straight line with the file's (descriptive) name in close proximity (e.g., the Analyst Report File in Fig. D-2). The direction of data flows going to or from a file is important, showing that a file only provides data to a process (outgoing arrow such as from the SAC-Interest-Installations file to the Edit Data process in Fig. D-2), only receives data from a process (incoming arrow such as from the Write Intelligence Report process to the Analyst Report file in Fig. D-2), or both. When there is two-way access between a file and a process, either a double-headed arrow or two separate arrows can be used to show the data flow depending on whether the data flows in question have the same composition. The rule is to only show net flows to and from files.

Sources and sinks are net originators or receivers of data which are outside the system's context and are represented by a named box (e.g., Intelligence Collection Agency in Fig. D-2). Data can flow both into and out of a source/sink box. They should be used sparingly since they are usually not defined very rigorously and are included to provide a feel for the system's connections to the outside world.

D.3 RULES FOR DRAWING DFDs (Rowell, 1981; pp. 63-69)

D.3.1 General Guidelines

1. Identify the system's net input and output data flows and draw them around the edge of the diagram. These data flows define the system's context boundary (since everything outside them is out of the system). Try to identify all of the important and relevant boundary data flows and include them, but do not be overly concerned with completeness at this point.

2. Work from inputs to out -- drawing in outputs, backwards from outputs to inputs, or from the center out drawing in processes and data flows. Concentrate first on major data flows looking for primary data pipelines. Draw them on the diagram with (for now) empty circles for the transforming processes and try to hook them up with the peripheral data flows. Examine each blank process and see if there are internal data flows which require the process to be split into two or more separate processes. For each data flow item, determine what is needed to build it, identify where the components come from, and try to determine what processes create the components.

3. Carefully label all the interface data flows. The names given to data flows will have a major impact on DFD readability. Therefore, try to make the name appropriate (applicable to the entire data flow, not just one of its components) and avoid grouping unlike items into a single flow unless they definitely belong together. Insure all data flows (except to and from simple files) are named. If the data flows cannot be simply and accurately named, consider: (1) breaking them up into two or more nameable data flows, or (2) restarting.

4. Label the processes in terms of their inputs and outputs. Again, make sure the name is appropriate and reflects what is done in the transformation. Try to develop names with a single strong action verb and a single object (multiple verbs usually mean more partitioning is required). Avoid verbs like "process" or "handle" - they are too imprecise. Repartition when necessary (to break down processes which are unnameable or to combine several processes to describe a process which is more easily named).

5. Ignore initialization or termination ideas. Draw the system in an up-and-running steady state and defer concerns about how the system got there until later (at the end of the entire DFD analysis process).

6. Ignore the details of trivial error-handling data flows.

> If the error requires no undoing of past processing, ignore it for the moment; if it requires you to back out previous updates or revert a file or files to a previous state, then do not ignore it (DeMarco, 1979; p. 68).

Remember that a DFD is trying to convey an overall picture of the system's context and contents -- leave the details until later.

7. Do not portray control flows or control information. There are two tests to determine if a data flow is in fact a control flow: (1) ask what data or information moves over the data flow in question -- if there is none, the data flow probably does not represent the stream of the data itself and should be removed; (2) ask what use the destination process will make of the data or information moving over the data flow in question.

   > If it is modified and put out as an outgoing data flow or part of one, then it is a legitimate data flow. If it only serves to prompt the process to start doing its work or guide it in how to do its work, then it is control (DeMarco, 1979; p. 69).

8. Be prepared to start the drawings over. The final DFD set should be preceded by several successively more accurate iterations.

D.3.2 Leveled DFDs

Leveled DFDs are called for when a system is too large or complex to be completely represented by a one-page DFD. Leveling involves first partitioning a system into subsystems, then treating each subsystem as a system which is partitioned into sub-subsystems, etc. until the required level of detail can be achieved for each of the lowest level processes.

The highest-level DFD (i.e., the system view) is referred to as the "parent," and lower-level DFDs (i.e., subsystem views) are "children," in the DeMarco DFD nomenclature. Insuring the equivalence of data flows between parent and child diagrams is called balancing.

D.3.2.1 Notational Conventions for Leveled DFDs

- Numbering -- Each child diagram retains the number of its parent's (related) process circles. Subprocesses are numbered in turn with decimal point separators. For example: Fig. D-3 below represents a more detailed (subsystem) view of higher-level processes which were pictured in Fig. D-2. The parent is "Diagram 0" and the child "Diagram 1." Process #1, "Edit Data" on Diagram 0, is decomposed into its constituent processes on Diagram 1, the child. Each subprocess on Diagram 1 is labeled as 1.1, 1.2, etc. If a process such as 1.2 were to be broken down further, the decomposition DFD would be labeled "Diagram 1.2" and the subprocesses 1.2.1, 1.2.2, etc.

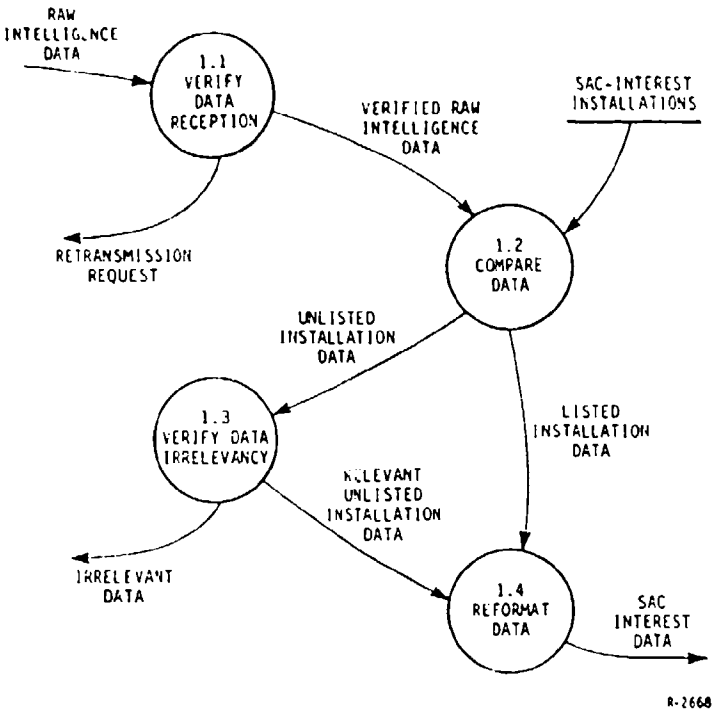


Figure D-3. Diagram 1: Data Flow Diagram of "Edit Data" Process (Rowell, 1981; p. 67)

D.3.2.2 Guidelines for Decomposition with Leveled DFDs

1. Asymmetry is allowed when trying to assess which processes should be decomposed. Some processes are likely to be more complex than others -- the more complex processes should be decomposed down more levels.

2. Stop decomposition when each lowest-leve. process can be specified by describing elemental tasks and procedures. Using IAT, one should be able to complete a frame by filling in values for slots at the point each lowest-level process has been adequately described.

3. Maintain accuracy and consistency in carrying out a leveled-DFD analysis:

   - Show changes on both parent and child diagrams.
   - Insure that data flows balance.
   - Suspect processes or files with incoming but not outgoing data flows; such processes/files may actually be sinks or errors.

REFERENCES

Bachert, R.F., K.H. Evers, and P.R. Santucci, "SADT/SAINT Simulation Technique," Proc. 1981 National Aerospace and Electronics Conference, Dayton, Ohio, 1981.

Barr, A. and E.A. Feigenbaum, The Handbook of Artificial Intelligence Vol. I, William Kaufman, Los Altos, California, 1981.

Beer, J., Cybernetics and Management, John Wiley and Sons, New York, 1959.

Berne, E., The Structure and Dynamics of Organizations and Groups, J.B. Lipincott, 1963.

Blitz, A.L., J.L. Teele, R.R. Tenney, and L.C. Kramer, "Micro-Modeler User's Manual," TR-273, ALPHATECH, Inc., Burlington, Massachusetts, 1985.

Boehm, B., Software Engineering Economics, Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

Brooks, F.A., Jr., "Operational Sequence Diagrams," I.R.E. Trans. Human Factors in Electronics, Vol. 1, No. 33, March 1960.

Chubb, G.P., "SAINT, A Digital Simulation Language for the Study of Manned Systems," in J. Moraal, K. - F. Kraiss, (Eds.), Manned System Design: Methods, Equipment, and Applications, Plenum Press, New York, 1981.

Colter, M., R.A. Miller, and J.G. Wohl, "Integrated Analysis Techniques (IAT) for Application to Command, Control, and Communications Systems," TM-HU-566/000/00, System Development Corporation, Huntsville, Alabama, September 1982.

DeMarco, T., Structured Analysis and System Specification, Prentice-Hall, Englewood Cliffs, New Jersey, 1978.

Goode, H.H., and R.E. Machol, System Engineering: An Introduction to the Design of Large-Scale Systems, McGraw Hill Book Company, Inc., New York, 1957.

Gruesbeck, M.A., R.A. Miller, J.G. Wohl, D.L. Kleinman, and G.P. Chubb, "An Integrated Analysis Technique (IAT) for C^3 Operator Performance Evaluation," TR-175, ALPHATECH, Inc., Burlington, Massachusetts, February 1984.

Klein, G.A., and J. Weitzenfeld, "Improvement of Skills for Solving Ill-Defined Problems," Educational Psychology, 13, 1978, pp. 31-34.

Kleinrock, R., Queuing Systems Vols. I & II, Wiley, New York, 1976.

Kornfeld, J.R., "Specification and Preliminary Validation of IAT Methods: Executive Summary," TR-223, ALPHATECH, Inc., Burlington, Massachusetts, December 1984.

Kornfeld, J.R., G.P. Chubb, R.A. Miller, and J.G. Wohl, "Specification and Preliminary Validation of IAT Methods: FY84 Annual Report," TR-224, ALPHATECH, Inc., Burlington, Massachusetts, February 1985.

Larkin, J., J. McDermott, D. Simon, and H. Simon, "Models of Competence in Solving Physics Problems," Cognitive Science, Vol. 4, 1980, pp. 317-345.

Machiavelli, N., The Prince, 1537, Translation by L. Ricci, Mentor Books, 1952.

Miller, J.G., Living Systems, McGraw-Hill Book Company, Inc., New York, Chapter 5, 1978.

Minsky, M., "Minsky's Frame System Theory," in Theoretical Issues in Natural Language Processing, R.C. Schank and B.L. Nash-Webber (Eds.), MIT Press, Cambridge, Massachusetts, 1975.

Neustadt, R. and E. May, Thinking in Time: The Uses of History for Decision-makers, The Free Press, New York, 1986.

Pattipati, K.R., J.J. Shaw, J.C. Deckert, L.K. Beean, M.G. Alexandridis, and W.E. Lougee, "CONFIDANTE: A Computer-Based Design Aid for the Optimal Synthesis, Analysis, and Operation of Maintenance Facilities," Proceedings 1984 AUTOTESTCON, Washington, D.C., November 1984, pp. 390-404.

Peterson, J.L., Petri Net Theory and the Modeling of Systems, Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

Rasmussen, J., and W. Rouse, Human Detection and Diagnosis of System Failures, Plenum Press, New York, 1981.

Rowell, P.V., "Specifying Users' Requirements in the Context of Military Intelligence Related Computer systems," Report No. LSSR 53-81, Master's Thesis, School of Systems and Logistics, Air Force Institute of Technology, WPAFB, Ohio, September 1981.

SOFTECH, Inc., Structured Analysis and Design Technique, SOFTECH, Inc., Waltham, Massachusetts, 1978.

SOFTECH, Inc., "Integrated Computer-Aided Manufacturing (ICAM) Architecture, Part II, Vol. IV - Function Modeling ($IDEF_0$)," AFWAL-TR-4023, Wright-Patterson AFB, Ohio, June 1981.

Schank, R.C., and R.P. Abelson, Scripts, Plans, Goals, and Understanding, Erlbaum Associates, Hillsdale, New Jersey, 1977.

Shaw, J.J., and D.P. Bertsekas, "Optimal Scheduling of Large Hydrothermal Power Systems," IEEE Trans. on Power Apparatus and Systems, Vol. PAS-104, No. 2, February, 1985, pp. 286-294.

Tenney, R.R., "Systematic Evaluation of Command and Control Systems," TR-288, ALPHATECH, Inc., Burlington, Massachusetts, April 1986.

Warfield, J.N.,"Intent Structures," IEEE Trans. Systems, Man, and Cybernetics, SMC-3, No. 2, March 1973.

Wohl, J.G., D. Gootkind, and H. D'Angelo, "Measures of Merit for Command and Control," MTR-8217, The MITRE Corporation, Bedford, Massachusett, 1981.

Yager, R.R., "Linguistic Representation of Default Values in Frames," IEEE Trans. on Systems, Man, and Cybernetics, Vol. SMC-14, No. 4, July/August 1984.